#2 a/F 12/13/01

Attorney Docket No. 1075.1179

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:

Miki TAKAGI et al.

Application No.: Unassigned | Group Art Unit: Unassigned

Filed: November 12, 2001 | Examiner:

For: AUTOMATED HDL MODIFYING APPARATUS AND COMPUTER-READABLE RECORDING MEDIUM IN WHICH PROGRAM FOR AUTOMATICALLY MODIFYING HDL IS RECORDED

SUBMISSION OF CERTIFIED COPY OF PRIOR FOREIGN APPLICATION IN ACCORDANCE WITH THE REQUIREMENTS OF 37 C.F.R. § 1.55

Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

In accordance with the provisions of 37 C.F.R. § 1.55, the applicant(s) submit(s) herewith a certified copy of the following foreign application:

Japanese Patent Application No. 2001-189337

Filed: June 22, 2001

It is respectfully requested that the applicant(s) be given the benefit of the foreign filing date(s) as evidenced by the certified papers attached hereto, in accordance with the requirements of 35 U.S.C. § 119.

Respectfully submitted,
STAAS & HALSEY LLP

Date: November 12, 2001 By: _____
James D. Halsey, Jr.
Registration No. 22,729

700 11th Street, N.W., Ste. 500
Washington, D.C. 20001
(202) 434-1500

日　本　国　特　許　庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日
Date of Application: ２００１年　６月２２日

出願番号
Application Number: 特願２００１－１８９３３７

出願人
Applicant(s): 富士通株式会社

２００１年　８月１０日

特許庁長官
Commissioner,
Japan Patent Office 及川耕造

出証番号　出証特２００１－３０７２２５７

【書類名】　特許願

【整理番号】　0150589

【提出日】　平成13年 6月22日

【あて先】　特許庁長官殿

【国際特許分類】　G06F 17/50

【発明の名称】　ＨＤＬ自動修正装置およびＨＤＬ自動修正プログラム並びに同プログラムを記録したコンピュータ読取可能な記録媒体

【請求項の数】　5

【発明者】

【住所又は居所】　神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

【氏名】　▲高▼木　美紀

【発明者】

【住所又は居所】　神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

【氏名】　竹山　広治

【発明者】

【住所又は居所】　神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

【氏名】　野口　弘

【特許出願人】

【識別番号】　000005223

【氏名又は名称】　富士通株式会社

【代理人】

【識別番号】　100092978

【弁理士】

【氏名又は名称】　真田　有

【電話番号】　0422-21-4222

【手数料の表示】

【予納台帳番号】　007696

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　9704824

【プルーフの要否】　要

【書類名】　　明細書

【発明の名称】　ＨＤＬ自動修正装置およびＨＤＬ自動修正プログラム並びに同プログラムを記録したコンピュータ読取可能な記録媒体

【特許請求の範囲】

【請求項１】　ハードウェア記述言語（以下、ＨＤＬという）によって記述された回路設計情報（以下、ＨＤＬ記述という）を自動的に修正する装置であって、

修正対象のＨＤＬ記述の字句解析を行なうＨＤＬ字句解析手段と、

該ＨＤＬ字句解析手段による解析結果に基づいて該ＨＤＬ記述の構文解析を行ない、該ＨＤＬ記述を解析木形式の記述に変換するＨＤＬ構文解析手段と、

該ＨＤＬ構文解析手段による解析結果に基づいて該ＨＤＬ記述の意味論解析を行ない、該ＨＤＬ記述における代入文の右辺と左辺の変数の型が不一致になっている部分を意味論的文法違反箇所として検出する意味論的文法違反検出手段と、

前記代入文の右辺の変数の型を左辺の変数の型に変換する型変換関数を、型変換規則として定義する型変換テンプレートと、

該意味論的文法違反検出手段によって該意味論的文法違反箇所とみなされた代入文の右辺に対して、該型変換テンプレートに定義された該型変換関数を適用することにより、該意味論的文法違反箇所を正しい記述に修正する意味論的文法違反修正手段と、

該意味論的文法違反修正手段による修正を施された該ＨＤＬ記述の逆構文解析を行ない、該ＨＤＬ記述を前記解析木形式の記述から通常の記述形式に変換するＨＤＬ逆構文解析手段と、

該意味論的文法違反修正手段によって修正を施された箇所に、該修正に関するコメントを付与するコメント付与手段とをそなえて構成されたことを特徴とする、ＨＤＬ自動修正装置。

【請求項２】　文法違反ではないが回路設計上考慮すべき修正対象事項、および、該修正対象事項に対応する箇所の修正規則を定義する制御情報テンプレートと、

該ＨＤＬ構文解析手段による解析結果に基づき、該ＨＤＬ記述において、該修

正対象事項に対応する箇所を検出する修正対象事項検出手段と、

該修正対象事項検出手段によって検出された箇所を、該制御情報テンプレートに定義された該修正規則に従って修正する修正対象事項修正手段とをさらにそなえ、

該ＨＤＬ逆構文解析手段が、該意味論的文法違反修正手段および該修正対象事項修正手段による修正を施された該ＨＤＬ記述の逆構文解析を行なうとともに、

該コメント付与手段が、該意味論的文法違反修正手段および該修正対象事項修正手段によって修正を施された箇所に、該コメントを付与することを特徴とする、請求項１記載のＨＤＬ自動修正装置。

【請求項３】　ハードウェア記述言語（以下、ＨＤＬという）によって記述された回路設計情報（以下、ＨＤＬ記述という）を自動的に修正する装置であって、

修正対象のＨＤＬ記述の字句解析を行なうＨＤＬ字句解析手段と、

該ＨＤＬ字句解析手段による解析結果に基づいて該ＨＤＬ記述の構文解析を行ない、該ＨＤＬ記述を解析木形式の記述に変換するＨＤＬ構文解析手段と、

文法違反ではないが回路設計上考慮すべき修正対象事項、および、該修正対象事項に対応する箇所の修正規則を定義する制御情報テンプレートと、

該ＨＤＬ構文解析手段による解析結果に基づき、該ＨＤＬ記述において、該修正対象事項に対応する箇所を検出する修正対象事項検出手段と、

該修正対象事項検出手段によって検出された箇所を、該制御情報テンプレートに定義された該修正規則に従って修正する修正対象事項修正手段と、

該修正対象事項修正手段による修正を施された該ＨＤＬ記述の逆構文解析を行ない、該ＨＤＬ記述を前記解析木形式の記述から通常の記述形式に変換するＨＤＬ逆構文解析手段と、

該修正対象事項修正手段によって修正を施された箇所に、該修正に関するコメントを付与するコメント付与手段とをそなえて構成されたことを特徴とする、ＨＤＬ自動修正装置。

【請求項４】　ハードウェア記述言語（以下、ＨＤＬという）によって記述された回路設計情報（以下、ＨＤＬ記述という）を、コンピュータに自動修正さ

せるためのＨＤＬ自動修正プログラムであって、

修正対象のＨＤＬ記述の字句解析を行なうＨＤＬ字句解析手段、

該ＨＤＬ字句解析手段による解析結果に基づいて該ＨＤＬ記述の構文解析を行ない、該ＨＤＬ記述を解析木形式の記述に変換するＨＤＬ構文解析手段、

該ＨＤＬ構文解析手段による解析結果に基づいて該ＨＤＬ記述の意味論解析を行ない、該ＨＤＬ記述における代入文の右辺と左辺の変数の型が不一致になっている部分を意味論的文法違反箇所として検出する意味論的文法違反検出手段、

該意味論的文法違反検出手段によって該意味論的文法違反箇所とみなされた代入文の右辺に対して、前記代入文の右辺の変数の型を左辺の変数の型に変換する型変換関数を適用することにより、該意味論的文法違反箇所を正しい記述に修正する意味論的文法違反修正手段、

該意味論的文法違反修正手段による修正を施された該ＨＤＬ記述の逆構文解析を行ない、該ＨＤＬ記述を前記解析木形式の記述から通常の記述形式に変換するＨＤＬ逆構文解析手段、および、

該意味論的文法違反修正手段によって修正を施された箇所に、該修正に関するコメントを付与するコメント付与手段として、該コンピュータを機能させることを特徴とする、ＨＤＬ自動修正プログラム。

【請求項５】　ハードウェア記述言語（以下、ＨＤＬという）によって記述された回路設計情報（以下、ＨＤＬ記述という）を、コンピュータに自動修正させるためのＨＤＬ自動修正プログラムを記録したコンピュータ読取可能な記録媒体であって、

該ＨＤＬ自動修正プログラムが、

修正対象のＨＤＬ記述の字句解析を行なうＨＤＬ字句解析手段、

該ＨＤＬ字句解析手段による解析結果に基づいて該ＨＤＬ記述の構文解析を行ない、該ＨＤＬ記述を解析木形式の記述に変換するＨＤＬ構文解析手段、

該ＨＤＬ構文解析手段による解析結果に基づいて該ＨＤＬ記述の意味論解析を行ない、該ＨＤＬ記述における代入文の右辺と左辺の変数の型が不一致になっている部分を意味論的文法違反箇所として検出する意味論的文法違反検出手段、

該意味論的文法違反検出手段によって該意味論的文法違反箇所とみなされた代

入文の右辺に対して、前記代入文の右辺の変数の型を左辺の変数の型に変換する型変換関数を適用することにより、該意味論的文法違反箇所を正しい記述に修正する意味論的文法違反修正手段、

該意味論的文法違反修正手段による修正を施された該ＨＤＬ記述の逆構文解析を行ない、該ＨＤＬ記述を前記解析木形式の記述から通常の記述形式に変換するＨＤＬ逆構文解析手段、および、

該意味論的文法違反修正手段によって修正を施された箇所に、該修正に関するコメントを付与するコメント付与手段として、該コンピュータを機能させることを特徴とする、ＨＤＬ自動修正プログラムを記録したコンピュータ読取可能な記録媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、例えばＶＨＤＬ〔ＶＨＳＩＣ（Very High Speed Integrated Circuit）ＨＤＬ〕やVerilog-ＨＤＬなどのハードウェア記述言語〔以下、ＨＤＬ（Hardware Discription Language）という〕によって記述された、電子システムや論理回路の設計情報における、不適切な記述を自動修正するための装置およびプログラム並びに同プログラムを記録したコンピュータ読取可能な記録媒体に関する。

【０００２】

【従来の技術】

ＬＳＩ（Large Scale Integration）等の半導体集積回路を設計する際、その半導体集積回路の論理回路についての設計情報を、ＶＨＤＬやVerilog-ＨＤＬ等のＨＤＬによって記述することが行なわれている。ＨＤＬは、一般的なソフトウェアプログラミング言語とは異なるもので、電子システムや論理回路の機能や構造を記述するのに適し、設計レベルを論理ゲートレベルからマイクロアーキテクチャレベルに高め階層設計を行なうのに適している。そして、上述のごとくＨＤＬによって記述された設計情報（以下、ＨＤＬ記述という場合がある）に基づいて、論理回路（ネットリスト）が論理合成ツールにより自動的に合成される。

【０００３】

このような論理合成を行なう前、つまり、ＨＤＬ記述を論理合成ツールに入力する前には、ＨＤＬ記述における文法違反のチェック（構文チェック）が行なわれている。この文法違反のチェックに際しては、例えば下記公報に開示された技術を用いることができる。

【０００４】

例えば特開平２－２９４７３６号公報では、ソフトウェアプログラミング言語で記述されたソースプログラムにおいて、例えば文末に記入されるべきセミコロン“；”が抜けている等の、比較的軽度の構文的文法違反を検出し、その文法違反を自動修正する技術が開示されている。この技術を用いることにより、ソースプログラム中の文末に“；”が抜けている場合、その“；”が自動的に挿入される。

【０００５】

また、例えば特開平６－４４０８１号公報では、ＰＡＳＣＡＬ言語で記述された入力ソースプログラムについて、代入文の右辺と左辺における変数の型が不一致である場合に、その旨を通知するエラーメッセージを出力する技術が開示されている。

【０００６】

【発明が解決しようとする課題】

特開平２－２９４７３６号公報に開示された技術を用いてＨＤＬ記述の文法違反チェックを行なった場合、上述のような構文的文法違反（文法違反としては軽度なもの）については自動修正することはできるが、代入文の右辺と左辺における変数の型が一致しないといった、意味論的文法違反（文法違反としては重度なもの）を自動修正することができない。

【０００７】

従って、意味論的文法違反については、設計者（ユーザ）が、ＨＤＬ処理系から出力されるエラーメッセージを参照して手修正しなければならない。その際、エラーメッセージは、具体的にどのような修正を行なえば良いかを示すものではないため、１回の修正で意味論的文法違反を解消できない場合が多く、設計者に

かかる負担が大きいという課題があった。特に、上述のような意味論的文法違反は、比較的高い頻度で発生するため、その文法違反を修正するために設計者にかかる負担はかなり大きいものとなる。

【０００８】

また、ＨＤＬ記述が自動修正された場合、設計者は、自動修正結果が意図するものかどうか、即ち適正な修正が行なわれたか否かを確認する必要があるが、従来技術では、修正箇所を示す目印等が自動修正結果に含まれていないので、ＨＤＬ記述のどこをどのように修正したかを確認するのに手間がかかり、設計者にかかる負担が大きいという課題もあった。

【０００９】

特開平６－４４０８１号公報に開示された技術を用いてＨＤＬ記述の文法違反チェックを行なった場合も、設計者は、代入文の右辺と左辺における変数の型が不一致である意味論的文法違反の発生を、エラーメッセージによって認識することはできるが、その文法違反を自動的に修正することができない。また、設計者は、エラーメッセージにより文法違反箇所および違反内容を知ることができても、その文法違反にどのように対処したらよいか分からず、文法書等を参照して文法違反の解消手法を検討しながら手修正する必要があるため、設計者にかかる負担は極めて大きい。

【００１０】

一方、従来技術では、文法違反をチェックすることができても、文法違反ではないが回路設計を行なう上で考慮すべき箇所（不適切な記述）をチェックして自動的に適切な記述に修正することができなかった。

従来、文法違反ではないが回路設計を行なう上で考慮すべき、名前の命名規則や論理合成記述規則（論理合成が可能な記述に関する規則）などに違反する箇所を検出しその違反内容を出力するスタイルチェッカも提案されているが、このスタイルチェッカは、違反内容を出力するだけで、その違反を回避するようにＨＤＬ記述を自動修正することはできない。従って、設計者は、スタイルチェッカからのエラーメッセージを参照して手修正でその違反を解消しなければならず、設計者にかかる負担が大きい。特に、複数の設計者によりＨＤＬを用いて回路設計

を行なった場合などには、命名規則に違反する箇所が多数出現する可能性が高く、違反箇所の名前を修正するのに多大な労力を要することになる。

【００１１】

また、文法違反ではないが回路の階層設計上考慮すべき規則や配線の接続関係の規則に違反するＨＤＬ記述のいくつかは、従来、フロントエンド（言語処理系）ではなく、バックエンド（論理合成ツールや検証ツール）でないと検出できない。そのようなＨＤＬ記述は、設計者の単純ミスによるものであるにもかかわらず早期に発見できないため、当然、自動修正することもできず、設計工程の手戻りの発生要因となっている。例えば、ＨＤＬ記述が複数の階層から構成されている場合、各階層での端子定義記述とインスタンス部における端子記述とが不一致になることが多々ある。このような不一致は、ミスによって生じる場合もあるが、文法上あるいは回路表現上において全く問題ない場合もある。しかし、上記不一致は、後で回路設計上の問題を生じさせ上述のような手戻りの発生要因となるおそれがあるため、上記不一致を含まない記述にＨＤＬ記述を自動修正できるようにすることが望まれている。

【００１２】

さらに、ＨＤＬを用いて回路設計を行なった場合、システム間の連係や設計フローの都合などにより、当初用いられていたＨＤＬから異なる種類の他のＨＤＬにＨＤＬ記述を変換することがしばしばある。このとき、当然、ＨＤＬ記述が現在のＨＤＬの言語規約を満たしていても他のＨＤＬの言語規約を満たさない場合があるため、上述のような変換の可能性がある場合、他のＨＤＬに変換した時に回路設計上の問題が生じないように、予め他のＨＤＬの言語規約も満たすようにＨＤＬ記述を自動修正できるようにすることが望まれている。

【００１３】

また、論理合成したいＨＤＬ記述においては、論理検証時に使用した論理合成不能な波形観測用シミュレーション記述等がコメントアウトされずに残っている場合が多々ある。このような論理合成不能な記述についても、自動修正（自動削除）する手立てがなく、従来、設計者が、ＨＤＬ記述を参照し論理合成不能な記述を見つけた場合に手修正するか、論理合成ツールでその記述によってエラーが

発生してから何らかの対処をとるしかなく、いずれにしても設計者にとって大きな負担となっていた。

【００１４】

本発明は、このような課題に鑑み創案されたもので、重度の意味論的文法違反を自動修正し且つその修正箇所が明確になるようにするほか、文法違反ではないが回路設計上考慮すべき箇所を適切な記述に自動修正し且つその修正箇所が明確になるようにして、設計者にかかる負担を大幅に軽減するとともに、高品質のＨＤＬ記述が得られるようにした、ＨＤＬ自動修正装置およびＨＤＬ自動修正プログラム並びに同プログラムを記録したコンピュータ読取可能な記録媒体を提供することを目的とする。

【００１５】

【課題を解決するための手段】

上記目的を達成するために、本発明のＨＤＬ自動修正装置（請求項１）は、ＨＤＬで記述されたＨＤＬ記述（回路設計情報）を自動的に修正する装置であって、修正対象のＨＤＬ記述の字句解析を行なうＨＤＬ字句解析手段と、該ＨＤＬ字句解析手段による解析結果に基づいて該ＨＤＬ記述の構文解析を行ない該ＨＤＬ記述を解析木形式の記述に変換するＨＤＬ構文解析手段と、該ＨＤＬ構文解析手段による解析結果に基づいて該ＨＤＬ記述の意味論解析を行ない該ＨＤＬ記述における代入文の右辺と左辺の変数の型が不一致になっている部分を意味論的文法違反箇所として検出する意味論的文法違反検出手段と、前記代入文の右辺の変数の型を左辺の変数の型に変換する型変換関数を型変換規則として定義する型変換テンプレートと、該意味論的文法違反検出手段によって該意味論的文法違反箇所とみなされた代入文の右辺に対して、該型変換テンプレートに定義された該型変換関数を適用することにより該意味論的文法違反箇所を正しい記述に修正する意味論的文法違反修正手段と、該意味論的文法違反修正手段による修正を施された該ＨＤＬ記述の逆構文解析を行ない該ＨＤＬ記述を前記解析木形式の記述から通常の記述形式に変換するＨＤＬ逆構文解析手段と、該意味論的文法違反修正手段によって修正を施された箇所に該修正に関するコメントを付与するコメント付与手段とをそなえて構成されたことを特徴としている。

【００１６】

このとき、文法違反ではないが回路設計上考慮すべき修正対象事項および該修正対象事項に対応する箇所の修正規則を定義する制御情報テンプレートと、該ＨＤＬ構文解析手段による解析結果に基づき、該ＨＤＬ記述において該修正対象事項に対応する箇所を検出する修正対象事項検出手段と、該修正対象事項検出手段によって検出された箇所を該制御情報テンプレートに定義された該修正規則に従って修正する修正対象事項修正手段とをさらにそなえ、該ＨＤＬ逆構文解析手段が、該意味論的文法違反修正手段および該修正対象事項修正手段による修正を施された該ＨＤＬ記述の逆構文解析を行なうとともに、該コメント付与手段が、該意味論的文法違反修正手段および該修正対象事項修正手段によって修正を施された箇所に、該コメントを付与するように構成してもよい（請求項２）。

【００１７】

また、本発明のＨＤＬ自動修正装置（請求項３）は、ＨＤＬによって記述されたＨＤＬ記述を自動的に修正する装置であって、上述したＨＤＬ字句解析手段およびＨＤＬ構文解析手段をそなえるとともに、文法違反ではないが回路設計上考慮すべき修正対象事項および該修正対象事項に対応する箇所の修正規則を定義する制御情報テンプレートと、該ＨＤＬ構文解析手段による解析結果に基づき該ＨＤＬ記述において該修正対象事項に対応する箇所を検出する修正対象事項検出手段と、該修正対象事項検出手段によって検出された箇所を、該制御情報テンプレートに定義された該修正規則に従って修正する修正対象事項修正手段と、該修正対象事項修正手段による修正を施された該ＨＤＬ記述の逆構文解析を行ない該ＨＤＬ記述を前記解析木形式の記述から通常の記述形式に変換するＨＤＬ逆構文解析手段と、該修正対象事項修正手段によって修正を施された箇所に該修正に関するコメントを付与するコメント付与手段とをそなえて構成されたことを特徴としている。

【００１８】

さらに、本発明のＨＤＬ自動修正プログラム（請求項４）は、ＨＤＬによって記述されたＨＤＬ記述を、コンピュータに自動修正させるためのものであって、上述したＨＤＬ字句解析手段，ＨＤＬ構文解析手段，意味論的文法違反検出手段

，意味論的文法違反修正手段，ＨＤＬ逆構文解析手段およびコメント付与手段として、該コンピュータを機能させることを特徴としている。そして、本発明のコンピュータ読取可能な記録媒体（請求項５）は、上述したＨＤＬ自動修正プログラムを記録したものである。

【００１９】

【発明の実施の形態】

以下、図面を参照して本発明の実施の形態を説明する。

〔１〕本発明の一実施形態の構成の説明

図１は本発明の一実施形態としてのＨＤＬ自動修正装置の構成を示すブロック図であり、この図１に示すように、本実施形態のＨＤＬ自動修正装置１は、ＨＤＬによって記述されたＨＤＬ記述（回路設計情報）を自動的に修正するものであって、ＨＤＬ字句解析手段１１，ＨＤＬ構文解析手段１２，構文的文法違反検出手段１３，構文的文法違反修正手段１４，意味論的文法違反検出手段１５，意味論的文法違反修正手段１６，修正対象事項検出手段１７，修正対象事項修正手段１８，ＨＤＬ逆構文解析手段１９，コメント付与手段２０，データベース２１～２４およびテンプレート３０，４０，５１～５５をそなえて構成されている。

【００２０】

ＨＤＬ字句解析手段１１は、修正対象のＨＤＬ記述（初期ＨＤＬ記述）２Ａの字句解析を行なうもので、ＨＤＬ記述２Ａを文字列の基本単位（トークン）に分解し、その基本単位を各基本単位の種別に関する情報とともにトークンデータベース２１に書き込むものである。

ＨＤＬ構文解析手段１２は、トークンデータベース２１のトークン（ＨＤＬ字句解析手段１１による解析結果）に基づいて、ＨＤＬ記述２Ａの構文解析を行ない、ＨＤＬ記述２Ａを解析木形式の記述に変換するものである。

【００２１】

文法解析テンプレート３０は、予約語の使用規則や、スペルについての規則や、構文規則などを定義するものである。ここで、構文規則は、例えば文中に“＝”があった場合、その文末には“；”が記入されるといった構文的文法についての規則である。

構文的文法違反検出手段１３は、ＨＤＬ構文解析手段１２による解析結果と文法解析テンプレート３０とに基づいて、ＨＤＬ記述２Ａにおける構文的文法違反箇所（スペルの間違っている箇所や、“＝”を記述する文の文末に“；”が抜けている箇所や、間違って予約語を使用している箇所など）を検出するもので、構文的文法違反修正手段１４は、構文的文法違反検出手段１３によって検出された構文的文法違反箇所を、文法解析テンプレート３０によって定義された規則に従って、正しい記述に修正するものである。

【００２２】

この構文的文法違反修正手段１４による修正結果が、ＨＤＬデータベース２２に書き込まれるようになっている。なお、構文的文法違反検出手段１３によって構文的文法違反箇所が検出されなかった場合には、構文的文法違反修正手段１４による修正処理は実行されず、ＨＤＬ構文解析手段１２による解析結果がそのままＨＤＬデータベース２２に書き込まれることになる。

【００２３】

意味論的文法違反検出手段１５は、ＨＤＬデータベース２２のデータ（ＨＤＬ構文解析手段１２による解析結果もしくは構文的文法違反修正手段１４による修正結果）に基づいてＨＤＬ記述２Ａの意味論解析を行ない、ＨＤＬ記述２Ａにおける代入文の右辺と左辺の変数の型が不一致になっている部分を意味論的文法違反箇所として検出するものである。

【００２４】

型変換テンプレート４０は、各種代入文の右辺の変数の型を左辺の変数の型に変換する型変換関数を、型変換規則として定義するもので、その型変換関数については、表１を参照しながら後述する。

意味論的文法違反修正手段１６は、意味論的文法違反検出手段１５によって意味論的文法違反箇所とみなされた代入文の右辺に対して、型変換テンプレート４０に定義された型変換関数を適用することにより、意味論的文法違反箇所を正しい記述に修正するもので、その修正結果はＨＤＬデータベース２３に書き込まれるようになっている。本実施形態のＨＤＬ自動修正装置１による意味論的文法違反箇所の具体的な修正動作については、図２を参照しながら後述する。

【００２５】

なお、意味論的文法違反検出手段１５によって意味論的文法違反箇所が検出されなかった場合には、意味論的文法違反修正手段１６による修正処理は実行されず、ＨＤＬデータベース２２をそのままＨＤＬデータベース２３として用いることになる。

【００２６】

テンプレート（制御情報テンプレート）５１～５５は、それぞれ、文法違反ではないが回路設計上考慮すべき修正対象事項、および、その修正対象事項に対応する箇所の修正規則を定義するもので、その詳細については後述する。

修正対象事項検出手段１７は、ＨＤＬデータベース２３のデータ（ＨＤＬ構文解析手段１２による解析結果もしくは構文的文法違反修正手段１４／意味論的文法違反修正手段１６による修正結果）に基づき、ＨＤＬ記述２Ａの中で、テンプレート５１～５５のそれぞれで定義された修正対象事項に対応する箇所を検出するものである。

【００２７】

修正対象事項修正手段１８は、修正対象事項検出手段１７によって検出された箇所を、テンプレート５１～５５のそれぞれに定義された修正規則に従って修正するもので、その修正結果はＨＤＬデータベース２４に書き込まれるようになっている。

なお、修正対象事項検出手段１７によって修正対象事項に対応する箇所が検出されなかった場合、修正対象事項修正手段１８による修正処理は実行されず、ＨＤＬデータベース２３をそのままＨＤＬデータベース２４として用いることになる。

【００２８】

ＨＤＬ逆構文解析手段１９は、ＨＤＬデータベース２４のデータ、つまり、構文的文法違反修正手段１４，意味論的文法違反修正手段１６や修正対象事項修正手段１８による修正を施されたＨＤＬ記述（解析木形式の記述）についての逆構文解析を行ない、そのＨＤＬ記述を解析木形式の記述から通常の記述形式に変換し、その変換結果を修正後のＨＤＬ記述２Ｂとして出力するものである。

【００２９】

コメント付与手段２０は、修正箇所（構文的文法違反修正手段１４，意味論的文法違反修正手段１６や修正対象事項修正手段１８によって修正を施された箇所）に、その修正に関するコメントを付与するものである。このコメント付与手段２０によってＨＤＬ記述２Ｂに付与される具体的なコメント等については、図２～図９を参照しながら後述する。なお、本実施形態のＨＤＬ自動修正装置１によって初期ＨＤＬ記述２Ａに対する修正が何ら施されなかった場合、コメント付与手段２０は、当然、ＨＤＬ記述２Ｂに対し何らコメントを付与しない。また、コメント付与手段２０は、ＨＤＬ逆構文解析手段１９から出力されたＨＤＬ記述２Ｂ（ソース）にコメントを付与する。

【００３０】

ここで、上述したテンプレート５１～５５について、より詳細に説明する。

言語変換規則テンプレート５１における修正対象事項は、修正対象事項検出手段１７が、処理中のＨＤＬ（例えばＶＨＤＬ）を異なる種類の他のＨＤＬ（例えばVerilog-ＨＤＬ）に変換した場合に他のＨＤＬの言語規約を満たさない箇所を、修正対象事項に対応する箇所として検出するように定義されている。また、制御情報テンプレート５１における修正規則は、修正対象事項修正手段１８が、修正対象事項検出手段１７によって検出された箇所を、他のＨＤＬの言語規約を満たす記述に変換・修正するように定義されている。

【００３１】

ここで、言語変換規則テンプレート５１は、図３に示すような予約語テンプレート５１ａ，名前テンプレート５１ｂ，名前生成ルール５１ｃおよび大文字小文字ルール５１ｄを有している。予約語テンプレート５１ａには、各種ＨＤＬごとに使用可能な予約語が登録・定義され、名前テンプレート５１ｂには、各種ＨＤＬごとに使用可能な端子名やネット名が登録・定義されている。また、名前生成ルール５１ｃには、端子名やネット名を修正する際に、ＨＤＬ記述に存在する文字列と重複しないユニークな文字列を新たな端子名やネット名として生成するための規則が定義されている。さらに、大文字小文字ルール５１ｄには、各種ＨＤＬが、それぞれ、アルファベットの大文字と小文字を区別して使用するものであ

るか、もしくは、アルファベットの大文字と小文字を区別しないで使用するものであるかが登録・定義されている。

【００３２】

図３を参照しながら後述するごとく、例えば、予約語テンプレート５１ａおよび名前テンプレート５１ｂを用い、例えば、Ｖerilog-ＨＤＬによるＨＤＬ記述中に、Ｖerilog-ＨＤＬでは端子名やネット名として使用可能であるがＶＨＤＬでは予約語として使用される文字列が検出された場合、その文字列が、ＶＨＤＬによる記述を行なった場合に予約語に該当しないような、新たな文字列に変換・修正される。その際、名前生成ルール５１ｃを用いて、ＨＤＬ記述に存在する文字列と重複しないユニークな文字列が、新たな文字列として生成されるようになっている。

【００３３】

また、上述した大文字小文字ルール５１ｄ（図３参照）によって処理中のＨＤＬがアルファベットの大文字と小文字を区別して使用するものであると認識された場合（例えばＶerilog-ＨＤＬ）、処理中のＨＤＬを、大文字と小文字を区別しないで使用する他のＨＤＬに変換する可能性を考慮し、修正対象事項検出手段１７が、大文字と小文字を区別して記述された同一綴りの一対の文字列のうちの一方を、修正対象事項に対応する箇所として検出するように、言語変換規則テンプレート５１における修正対象事項を定義してもよい。その際、言語変換規則テンプレート５１における修正規則は、修正対象事項修正手段１８が、上述の名前生成ルール５１ｃに従って、ＨＤＬ記述中に存在する文字列の綴りと重複しない綴りの新しい文字列を生成してから、修正対象事項検出手段１７によって検出された一方の文字列を新しい文字列に置換するように定義される。

【００３４】

逆に、上述した大文字小文字ルール５１ｄ（図３参照）によって処理中のＨＤＬがアルファベットの大文字と小文字を区別しないで使用するものであると認識された場合（例えばＶＨＤＬ）、処理中のＨＤＬを、大文字と小文字を区別して使用する他のＨＤＬに変換する可能性を考慮し、修正対象事項検出手段１７が、ＨＤＬ記述の文字列における大文字および小文字のうちの一方を、修正対象事項

に対応する箇所として検出するように、言語変換規則テンプレート５１における修正対象事項を定義してもよい。その際、修正対象事項修正手段１８が、修正対象事項検出手段１７によって検出された大文字または小文字をそれぞれ小文字または大文字に変換するように、言語変換規則テンプレート５１における修正規則が定義される。

なお、上述した言語変換規則テンプレート５１を用いた具体的な修正動作については、図３を参照しながら後述する。

【００３５】

使用禁止文字情報テンプレート５２における修正対象事項は、修正対象事項検出手段１７が、所定の使用禁止文字を含む文字列を、修正対象事項に対応する箇所として検出するように定義されている。また、使用禁止文字情報テンプレート５２における修正規則は、修正対象事項修正手段１８が、ＨＤＬ記述に存在する文字列と重複せず且つ所定の使用禁止文字を含まない新しい文字列を生成してから、修正対象事項検出手段１７によって検出された使用禁止文字を含む文字列を新しい文字列に置換するように定義されている。

【００３６】

ここで、使用禁止文字情報テンプレート５２は、図４に示すような使用禁止文字テンプレート５２ａおよび名前生成ルール５２ｂを有している。使用禁止文字テンプレート５２ａには、使用を禁止されている文字列が定義・登録されており、修正対象事項検出手段１７は、使用禁止文字テンプレート５２ａを参照して使用禁止文字を含む文字列を、修正対象事項に対応する箇所として検出するようになっている。また、名前生成ルール５２ｂは、前述した名前生成ルール５１ｃと同様のもので、ＨＤＬ記述に存在する文字列と重複せず且つ所定の使用禁止文字を含まないユニークな文字列を新たに生成するための規則が定義されており、修正対象事項修正手段１８は、名前生成ルール５２ｂに従って、上述した新しい文字列を生成している。

なお、上述した使用禁止文字情報テンプレート５２を用いた具体的な修正動作については、図４を参照しながら後述する。

【００３７】

階層情報テンプレート５３における修正対象事項は、修正対象事項検出手段１７が、ＨＤＬ記述を成す複数の階層において端子記述が不一致もしくは不統一である箇所を、修正対象事項に対応する箇所として検出するように定義されている。また、階層情報テンプレート５３における修正規則は、修正対象事項修正手段１８が、修正対象事項検出手段１７によって検出された箇所における端子記述を、複数の階層の全てにおいて一致または統一した記述に修正するように定義されている。

【００３８】

なお、上述した階層情報テンプレート５３を用いた具体的な修正動作については、図５～図８を参照しながら後述する。また、階層情報テンプレート５３は、例えば、図５～図８のそれぞれに示すような修正規則５３ａ～５３ｄを有している。これらの修正規則５３ａ～５３ｄの詳細については後述する。

【００３９】

接続情報テンプレート５４の修正対象事項は、修正対象事項検出手段１７が、信号代入記述の左辺と右辺との関係が誤っている箇所を、修正対象事項に対応する箇所として検出するように定義されている。また、接続情報テンプレート５４の修正規則は、修正対象事項修正手段１８が、修正対象事項検出手段１７によって検出された箇所における信号代入記述の左辺と右辺との関係を修正するように定義されている。なお、上述した接続情報テンプレート５４を用いた具体的な修正動作については後述する。

【００４０】

非合成記述テンプレート５５の修正対象事項は、修正対象事項検出手段１７が、論理合成ツールが合成することのできない箇所（非合成記述箇所）を、修正対象事項に対応する箇所として検出するように定義されている。また、非合成記述テンプレート５５の修正規則は、修正対象事項修正手段１８が修正対象事項検出手段１７によって検出された箇所を削除するように、もしくは、修正対象事項修正手段１８が、修正対象事項検出手段１７によって検出された箇所に、当該箇所における記述を論理合成ツールに無視させるためのディレクティブを追加・記入するように、定義される。設計者は、非合成記述箇所を削除する修正規則とディ

レクティブを追加・記入する修正規則とのいずれか一方を選択して採用することができるようになっている。

【００４１】

なお、上述した非合成記述テンプレート５５を用いた具体的な修正動作については、図９を参照しながら後述する。また、非合成記述テンプレート５５は、図９に示すような修正規則５５ａを有している。この修正規則５５ａの詳細については後述するが、図９に示す修正規則５５ａは、ディレクティブを追加・記入するように定義されたものである。

【００４２】

上述したＨＤＬ字句解析手段１１，ＨＤＬ構文解析手段１２，構文的文法違反検出手段１３，構文的文法違反修正手段１４，意味論的文法違反検出手段１５，意味論的文法違反修正手段１６，修正対象事項検出手段１７，修正対象事項修正手段１８，ＨＤＬ逆構文解析手段１９およびコメント付与手段２０は、専用ソフトウェア（ＨＤＬ自動修正プログラム）によって実現される。

【００４３】

このＨＤＬ自動修正プログラムは、例えばフレキシブルディスク，ＣＤ－ＲＯＭ等のコンピュータ読取可能な記録媒体に記録された形態で提供される。本実施形態のＨＤＬ自動修正装置１は、ＣＰＵ，ＲＯＭ，ＲＡＭなどから構成されたコンピュータ（図示省略）によって実現される。そして、ＲＯＭ等にＨＤＬ自動修正プログラムを予め格納しておき、このＨＤＬ自動修正プログラムを、ＣＰＵによって読み出し実行することにより、上述した各手段１１～２０としての機能が実現される。

【００４４】

なお、ＨＤＬ自動修正プログラムは、例えば磁気ディスク，光ディスク，光磁気ディスク等の記憶装置（記録媒体）に記録しておき、その記憶装置から通信経路を介してコンピュータに提供されてもよい。

また、テンプレート３０，４０，５１～５５についての情報は、設計者の手によりキーボード，マウス等を通じて入力してもよいし、別途、記録媒体を通じて入力してもよいし、上述したＨＤＬ自動修正プログラムに予め含めた形で提供さ

れてもよい。

さらに、上述したデータベース２１～２４としては、上記ＲＡＭを用いてもよいし、例えばフレキシブルディスク，ＣＤ－Ｒ，ＣＤ－ＲＷ等の記録媒体を用いてもよい。

【００４５】

〔２〕本発明の一実施形態の動作の説明

次に、上述のごとく構成された本実施形態のＨＤＬ自動修正装置１の動作について、より具体的に説明する。

まず、本実施形態のＨＤＬ自動修正装置１による一連の自動修正動作について簡単に説明する。

【００４６】

修正対象のＨＤＬ記述（初期ＨＤＬ記述）２Ａは、ＨＤＬ自動修正装置１に入力されると、まず、ＨＤＬ字句解析手段１１によりトークンに分解され、トークンデータベース２１に書き込まれる。このトークンに基づいて、ＨＤＬ構文解析手段１２により、ＨＤＬ記述２Ａの構文解析が行なわれ、ＨＤＬ記述２Ａが解析木形式の記述に変換される。得られた解析木と、文法解析テンプレート３０とに基づいて、構文的文法違反検出手段１３により、ＨＤＬ記述２Ａにおける構文的文法違反箇所が検出される。構文的文法違反箇所が検出された場合、その構文的文法違反箇所は、構文的文法違反修正手段１４により、文法解析テンプレート３０で定義された規則に従って、正しい記述に修正される。その修正結果は、ＨＤＬデータベース２２に書き込まれる。

【００４７】

ついで、意味論的文法違反検出手段１５により、上述のごとく構文的文法違反箇所を修正されたＨＤＬ記述２Ａの意味論解析が、ＨＤＬデータベース２２のデータに基づいて行なわれ、ＨＤＬ記述２Ａにおける代入文の右辺と左辺の変数の型が不一致になっている部分が意味論的文法違反箇所として検出される。意味論的文法違反箇所が検出された場合、意味論的文法違反修正手段１６により、その意味論的文法違反箇所における代入文の右辺に対して、型変換テンプレート４０に定義された型変換関数が適用される。これにより、意味論的文法違反箇所が正

しい記述に修正され、その修正結果がＨＤＬデータベース２３に書き込まれる。

【００４８】

さらに、修正対象事項検出手段１７により、上述のごとく意味論的文法違反箇所を修正されたＨＤＬ記述２Ａの中において、テンプレート５１～５５のそれぞれで定義された修正対象事項に対応する箇所が検出される。そのような箇所が検出された場合、修正対象事項修正手段１８により、その箇所が、テンプレート５１～５５のそれぞれに定義された修正規則に従って修正され、その修正結果がＨＤＬデータベース２４に書き込まれる。

【００４９】

以上のようにして各種修正を施されデータベース２４に保持された、解析木形式のＨＤＬ記述は、ＨＤＬ逆構文解析手段１９により、通常の記述形式に変換され、修正後のＨＤＬ記述２Ｂとして出力される。そして、そのＨＤＬ記述２Ｂの修正箇所には、コメント付与手段２０により、その修正に関するコメントが付与される。

【００５０】

さて、次に、図２～図９を参照しながら、ＶＨＤＬやVerilog-ＨＤＬによって記述されたＨＤＬ記述を修正対象として、本実施形態のＨＤＬ自動修正装置１が実際に行なう自動修正動作について説明する。ここでは、本発明の特徴的な部分である、意味論的文法違反や修正対象事項と、それらに対する修正動作と、修正結果に付与されるコメントとについて、具体的に説明する。

【００５１】

〔２－１〕意味論的文法違反の修正動作について

ＨＤＬ記述２Ａには、ＶＨＤＬのように代入文の右辺と左辺の型が不一致の場合に文法違反となるものがある。本実施形態では、そのような意味論的文法違反箇所を、意味論的文法違反検出手段１５によって検出すると、型変換テンプレート４０を用いて意味論的文法違反修正手段１６により自動的に修正した上で、その修正内容が、コメント付与手段２０により、ソース（修正後ＨＤＬ記述２Ｂ）の修正箇所にコメントとして追加・記入される。

【００５２】

例えば図２に示す初期ＨＤＬ記述（ＶＨＤＬによる記述）２Ａにおいては、入力変数“ａ”（右辺）の型“std_ulogic”と、出力変数“ｂ”（左辺）の型“bit”とは異なるものであるため、初期ＨＤＬ記述２Ａ中の“ｂ<=ａ；”の箇所が、意味論的文法違反箇所として検出される。上述のような型の不一致箇所は、意味論的文法違反検出手段１５がＨＤＬデータベース２２を検索することによって検出される。

【００５３】

そして、意味論的文法違反修正手段１６は、その箇所を修正するために必要な型変換パターン（型変換規則，型変換関数）が型変換テンプレート４０に存在するか否かをチェックする。ここで、型変換テンプレート４０においては、左辺の変数の型が“bit”で右辺の変数の型が“std_ulogic”である場合の型変換パターン（型変換規則）“To_bit”が、ライブラリ“std_logic_1164”によって提供・定義されている。

【００５４】

従って、意味論的文法違反箇所“ｂ<=ａ；”は、意味論的文法違反修正手段１６により、修正後ＨＤＬ記述２Ｂにおいて、“ｂ<=To_bit(ａ)；”に自動的に変換され修正される。さらに、その修正箇所“ｂ<=To_bit(ａ)；”の後には、コメント付与手段２０により、修正内容を示すコメント“--型変換”が付与される。

なお、型変換テンプレート４０には、典型的な型変換パターン（型変換規則，型変換関数）が予め定義されている。その型変換パターンの代表的なものを下記表１に示す。また、必要に応じ、型変換関数が格納されているライブラリとパッケージとを参照するためのlibrary文とuse文の追加を自動的に行なってもよい。

【００５５】

【表１】

左辺	右辺	ライブラリ	パッケージ
bit	std_ulogic	std_logic_1164	To_bit
bit_vector	std_logic_vector	std_logic_1164	To_bitvector
	std_ulogic_vector	std_logic_1164	To_bitvector
std_ulogic	bit	std_logic_1164	To_StdULogic
std_ulogic_vector	std_logic_vector	std_logic_1164	To_StdULogicVector
	bit_vector	std_logic_1164	To_StdULogicVector
std_logic_vector	bit_vector	std_logic_1164	To_StdLogicVector
	std_ulogic_vector	std_logic_1164	To_StdLogicVector
	integer	std_logic_arith	CONV_STD_LOGIC_VECTOR
	unsigned	std_logic_arith	CONV_STD_LOGIC_VECTOR
	signed	std_logic_arith	CONV_STD_LOGIC_VECTOR
	std_ulogic	std_logic_arith	CONV_STD_LOGIC_VECTOR
boolean	std_ulogic	std_logic_1164	Is_X
	std_logic_vector	std_logic_1164	is_X
	std_ulogic_vector	std_logic_1164	is_X
natural	unsigned	numeric_std	TO_INTEGER
integer	ｓｉｇｎｅｄ	numeric_std	TO_INTEGER
	std_logic_vector	std_logic_signed	CONV_INTEGER
	unsigned	std_logic_arith	CONV_INTEGER
	signed	std_logic_arith	CONV_INTEGER
	std_ulogic	std_logic_arith	CONV_INTEGER
unsigned	natural	numeric_std	TO_UNSIGNED
	integer	std_logic_arith	CONV_UNSIGNED
	signed	std_logic_arith	CONV_UNSIGNED
	std_ulogic	std_logic_arith	CONV_UNSIGNED
signed	ｎａｔｕａｌ	numeric_std	TO_SIGNED
	integer	std_logic_arith	CONV_SIGNED
	unsigned	std_logic_arith	CONV_SIGNED
	std_ulogic	std_logic_arith	CONV_SIGNED

【００５６】

〔２－２〕言語変換規則テンプレートを用いた修正動作について

ＨＤＬを用いて回路設計を行なった場合、システム間の連係や設計フローの都合などにより、当初用いられていたＨＤＬから異なる種類の他のＨＤＬにＨＤＬ記述を変換することがしばしばある。このとき、当然、ＨＤＬ記述が現在のＨＤＬの言語規約を満たしていても他のＨＤＬの言語規約を満たさない場合がある。本実施形態のＨＤＬ自動修正装置１では、上述のような変換の可能性を考慮して（他言語に変換した場合に起こりうる文法エラーを想定して）、他のＨＤＬに変換した時にも回路設計上の問題（他言語に変換した場合の文法エラー）が生じないように、予め他のＨＤＬの言語規約も満たすようにＨＤＬ記述を自動修正している。

【００５７】

つまり、実際に他言語へ変換するツールは存在しているが、本実施形態では、他言語への変換処理は行なわれないが、他言語へ変換されることを考慮した他言語の言語規約チェックが行なわれ、そのチェック結果に応じた自動修正が行なわれる。このように他言語での規約チェックを事前に行なうことで、設計者に負担をかけることなく、複数種類のＨＤＬにおいて回路設計上の問題を生じさせることのないＨＤＬ記述２Ｂを得ることができる。従って、いつでも他言語への変換が可能となり、また実際に他言語に変換した場合にエラー修正の手間を省くことができる。

【００５８】

そこで、本実施形態のＨＤＬ自動修正装置１における修正対象事項検出手段１７では、図３に示すように、言語変換規則テンプレート５１に含まれる予約語テンプレート５１ａ，名前テンプレート５１ｂおよび大文字小文字ルール５１ｄを読み込み、これらのテンプレート５１ａ，５１ｂや大文字小文字ルール５１ｄに違反している箇所をチェックする。そして、違反箇所における記述は、名前生成ルール５１ｃに従って自動的に生成された新たな記述（文字列）に変換される。また、他言語へ変換した場合の言語規約違反だけではなく、混乱を招きやすい記述に対しても、チェックおよび自動修正を行なう。

【００５９】

例えばVerilog-ＨＤＬで記述されたＨＤＬ記述２Ａに対しては、構文的文法違反検出手段１３によってVerilog-ＨＤＬの予約語チェックを行なうほか、修正対象事項検出手段１７によって、予約語テンプレート５１ａおよび名前テンプレート５１ｂを用いて、他のＨＤＬ、例えばＶＨＤＬの予約語チェックが行なわれる。このとき、予約語テンプレート５１ａには、変換可能性のある言語（ＨＤＬ）の予約語が全て定義・登録されており、名前テンプレート５１ｂには、変換可能性のある言語（ＨＤＬ）ごとに使用可能な端子名やネット名が登録・定義されている。

【００６０】

図３に具体的に示すVerilog-ＨＤＬによるＨＤＬ記述２Ａにおいて、“in”や“out”は、Verilog-ＨＤＬでは端子名やネット名として使用可能であるが、

ＶＨＤＬでは予約語となっているため、ＶＨＤＬでは、“in”や“out”を端子名やネット名として使用することは禁止されている。このように他言語に変換した場合にエラーとなる記述“in”や“out”が、修正対象事項検出手段１７により修正対象の箇所として検出されると、修正対象事項修正手段１８は、サフィックス，プレフィックス，連結子，追い番を記述した名前生成ルール５１ｃを読み込み、そのルール５１ｃに従って、新たな文字列（名前）として“in_1”や“out_1”をそれぞれ自動生成する。

【００６１】

ここで生成された新たな文字列“in_1”や“out_1”は、いずれも、ＨＤＬ記述２Ａに存在する文字列と重複しないユニークな文字列であり、且つ、Ｖerilog-ＨＤＬやＶＨＤＬのいずれにおいても端子名やネット名として使用可能なものである。

そして、“in”や“out”は、修正後ＨＤＬ記述２Ｂにおいて、それぞれ“in_1”や“out_1”に自動的に置き換えられ、さらに、その修正箇所の行末には、コメント付与手段２０により、修正コメント“//修正”が付与される。

【００６２】

また、例えばVerilog-ＨＤＬによるＨＤＬ記述２Ａにおいて、ネット名中の文字として“a _ _b”（図３参照）や“$”を使用することは、Ｖerilog-ＨＤＬの言語規約には違反していないが、ＶＨＤＬの言語規約に違反している。そのため、ＨＤＬ記述２ＡをVerilog-ＨＤＬからＶＨＤＬに変換した場合にエラーが発生するものと予測されるので、修正対象事項修正手段１８は、名前生成ルール５１ａに従って、上述と同様のユニークな文字列（名前）を自動生成し、他言語に変換した場合、違反するであろうと判断した箇所を自動発生名に変更する。

図３において、ＨＤＬ記述２Ａの文字列“a_ _b”は、修正後ＨＤＬ記述２Ｂでは、新たな文字列“a_b”に自動的に置き換えられ、さらに、その修正箇所の行末に、コメント付与手段２０により、修正コメント“//修正”が付与される。

【００６３】

さらに、Ｖerilog-ＨＤＬでは、アルファベットの大文字と小文字を区別して使用するが、ＶＨＤＬでは、アルファベットの大文字と小文字を区別しないで使

用する。このため、Verilog-ＨＤＬによるＨＤＬ記述２Ａにおいて、大文字と小文字とが混在している場合、そのＨＤＬ記述２ＡをVerilog-ＨＤＬからＶＨＤＬに変換した場合、混乱を招いてしまう。図３に示すＨＤＬ記述２Ａにおいて“ａ”と“Ａ”とは区別して使用されているが、ＶＨＤＬでは、これらの文字“ａ”および“Ａ”を区別することができず混乱を招くことになる。

【００６４】

そこで、本実施形態のＨＤＬ自動修正装置１では、Verilog-ＨＤＬをＶＨＤＬに変換する可能性を考慮し、修正対象事項検出手段１７によって、大文字と小文字を区別して記述された同一綴りの一対の文字列“Ａ”と“ａ”のうちの一方（ここでは“Ａ”）が、修正対象事項に対応する箇所として検出され、さらに、修正対象事項修正手段１８が、名前生成ルール５１ｃに従って、上述と同様のユニークな文字列（名前）として例えば“Ａ_1”を自動生成する。ここで生成された新たな文字列は、ＨＤＬ記述中に存在する文字列の綴りと重複しない綴りを有するものである。

【００６５】

そして、ＨＤＬ記述２Ａの文字“Ａ”は、修正後ＨＤＬ記述２Ｂにおいて、新たな文字列“Ａ_1”に自動的に置き換えられ、さらに、その修正箇所の行末に、コメント付与手段２０により、修正コメント“//修正”が付与される。このように“Ａ”を“Ａ_1”に置き換えることにより、Verilog-ＨＤＬにおいてもＶＨＤＬにおいても、その名前を区別することが可能になる。

【００６６】

逆に、アルファベットの大文字と小文字を区別しないＶＨＤＬで記述されたＨＤＬ記述をチェックする場合には、大文字と小文字を区別して使用するVerilog-ＨＤＬに変換する可能性を考慮し、修正対象事項検出手段１７によって、ＨＤＬ記述の文字列における大文字および小文字のうちの一方が、修正対象事項に対応する箇所として検出され、さらに、修正対象事項修正手段１８が、検出された大文字または小文字をそれぞれ小文字または大文字に自動的に変換する。即ち、ＶＨＤＬで記述されたＨＤＬ記述中の全ての文字を、小文字もしくは大文字に統一する修正が行なわれる。

【００６７】

ＶＨＤＬでは、例えば端子名“ｂ”と“Ｂ”とを同一の端子名として取り扱うが、Ｖｅｒｉｌｏｇ-ＨＤＬでは、これらの端子名“ｂ”と“Ｂ”とを異なる端子名として取り扱うため、混乱を招くことになるが、本実施形態のＨＤＬ自動修正装置１では、上述のごとく、ＶＨＤＬによるＨＤＬ記述２Ａにおける文字が全て小文字もしくは大文字に統一され、上述の端子名も全て“ｂ”もしくは“Ｂ”に置き換えられる。従って、上述のような混乱を招くこともなくなる。

【００６８】

〔２－３〕使用禁止文字情報テンプレートを用いた修正動作について

ＨＤＬを用いて回路設計を行なう際には、複数人数で設計を行なったり、ツールを使用して設計を行なったりする場合がある。このような場合、ＨＤＬ記述中の名前ルールが統一されていないことがしばしばある。そのため、従来、ツール間でインターフェイスする場合や他言語に変換する場合を考慮し、手作業により名前の記述の統一をはかっていた。

【００６９】

これに対し、本実施形態のＨＤＬ自動修正装置１の修正対象事項検出手段１７では、図４に示すように、使用禁止文字情報テンプレート５２に含まれる使用禁止文字テンプレート５２ａを読み込み、使用禁止文字テンプレート５２ａを用いて、ＨＤＬ記述２Ａの使用禁止文字を使用している箇所をチェックする。そして、モジュール名，外部端子名，インスタンス名，ネット名，タイプ名，コンポーネント名，外部端子別名等において使用禁止文字が使用されていた場合、その使用禁止文字を含む文字列（名前）は、名前生成ルール５２ｂに従って自動的に生成された新たな名前（文字列）に変換される。

【００７０】

図４では、使用禁止文字として例えば“＄”，“＆”を登録された使用禁止文字テンプレート５２ａを用いることで、修正対象事項検出手段１７により、入力Ｖｅｒｉｌｏｇ-ＨＤＬで記述されたＨＤＬ記述２Ａ中で“＄”，“＆”が使用されている箇所があるかどうかがチェックされる。このとき、図４に示す例では、インスタンス名として“＄”が修正対象箇所として検出され、修正対象事項修正手段

１８が、名前生成ルール５２ｂ（ここでは“prefix:X;”）に従って、ＨＤＬ記述２Ａに存在する文字列と重複せず且つ所定の使用禁止文字を含まない新しい文字列（名前）として、例えば“X1”を自動生成する。

【００７１】

そして、ＨＤＬ記述２Ａの文字“＄”は、修正後ＨＤＬ記述２Ｂにおいて、新たな文字列“X1”に自動的に置き換えられ、さらに、その修正箇所の行末に、コメント付与手段２０により、修正コメント“//修正”が付与される。

これにより、例えば複数の設計者によりＨＤＬ記述２Ａを作成した場合に命名規則に違反する箇所が多数出現したとしても、設計者に負担をかけることなく、その違反箇所の名前（文字列）を容易かつ確実に修正して命名規則に従ったＨＤＬ記述２Ｂを得ることができる。

【００７２】

〔２－４〕階層情報テンプレートを用いた修正動作について

ＨＤＬ記述２Ａが複数の階層から構成されている場合（例えば図５参照）、各階層の端子定義記述とインスタンス部における端子記述とが不一致（不統一）であることが多々ある。このような端子記述の不一致は、設計者が文法違反の記述を行なって生じる場合や、設計者が文法的には違反していないが回路として不適切な記述をしてしまって生じる場合があり、これらの場合は、いずれもその不一致を解消することが望ましい。

【００７３】

また、Verilog-ＨＤＬは、端子記述等の省略が許容される言語であるため、Verilog-ＨＤＬによるＨＤＬ記述中には、上述のような端子記述の不一致が多々ある。このような場合、上述のような不一致は回路表現として間違いではないが、安全な運用のためにできるだけ不一致の無い明快な記述を行なった方が好ましい。

【００７４】

そこで、本実施形態のＨＤＬ自動修正装置１では、

(1)全てのモジュールポート名（端子名）をインスタンス部に記述する（図５参照）。

(2)上位階層におけるビット幅の記述を、下位階層におけるビット幅の記述に合わせる（図６参照）。

(3)上位階層におけるコンポーネントポートの記述を下位階層の記述に合わせる（図７参照）。

(4)インスタンス部におけるポート名（端子名）を、名前対応もしくは位置対応で統一的に記述する（図８参照）。

というルールを、それぞれ階層情報テンプレート５３における修正規則５３ａ～５３ｄとして定義する。

【００７５】

このような修正規則５３ａ～５３ｄに従って、修正対象事項検出手段１７により、ＨＤＬ記述を成す複数の階層において端子記述（ポート名記述）が不一致もしくは不統一である箇所が、修正対象事項に対応する箇所として検出される。そして、修正対象事項修正手段１８により、検出された箇所における端子記述が、言語文法および修正規則５３ａ～５３ｄを定義する制御情報（階層情報テンプレート５３）に基づき、複数の階層の全てにおいて一致または統一した記述に自動的に修正される。

【００７６】

階層構造をもつVerilog-ＨＤＬによるＨＤＬ記述２Ａでは、インスタンス部に、未使用の下位階層端子名を記述する必要はない。しかし、Verilog-ＨＤＬからＶＨＤＬへの変換を行なう場合や、端子が未使用であることを明示したい場合には、下位階層の全ての端子名を記述した方が都合が良いので、全ての端子名（ポート名）をインスタンス部に記述するという修正規則５３ａ（verilog instance port:adjust to module port;）を、階層情報テンプレート５３に制御情報として定義・登録する。

【００７７】

図５に示すＨＤＬ記述２Ａでは、下位階層において端子名としてs,t,u,vの４つが記述されているが、上位階層では端子名ｖをもつ端子が使用されていないので、インスタンス部においてその端子名ｖの記述が省略されている。このようなＨＤＬ記述２ＡをＨＤＬ自動修正装置１に入力すると、ＨＤＬ記述２Ａのインス

タンス部における省略記述“ins(.s(a),.t(b),.u(c))”が、修正後ＨＤＬ記述２Ｂにおいて、全端子名を有する記述“ins(.s(a),.t(b),.u(c),.v())”に自動的に修正され、さらに、その修正箇所の行末に、コメント付与手段２０により、修正コメント“//修正”が付与される。これにより、ＨＤＬ記述２Ａ中のインスタンス部の記述が全端子名を有する記述に変更されるので、未使用の端子を明示することができるほか、Ｖerilog-ＨＤＬからＶＨＤＬへの変換にも対応することができるようになる。

【００７８】

図６に示すＶerilog-ＨＤＬによるＨＤＬ記述２Ａでは、下位階層において出力ｕは２ビット［0:1］であることが記述されているが、上位階層では出力ｕに対応する出力ｃが１ビットであり、インスタンス部において出力ｕのうちの未使用の１ビットの記述が省略されている。

【００７９】

このとき、ビット幅が不一致であった場合に上位階層のインスタンス部を下位モジュールのビット幅に合わせるという修正規則５３ｂ（bundle port:adjust to lower module）を、階層情報テンプレート５３に制御情報として定義・登録しておけば、図６に示すＨＤＬ記述２Ａは、インスタンス部における上位階層のビット幅の記述を下位階層のビット幅の記述に一致させたＨＤＬ記述２Ｂに、自動的に修正され、さらに、その修正箇所の行末に、コメント付与手段２０により修正コメント“//修正”が付与される。これにより、上位階層において、出力ｕの２ビットのうち１ビットが空いていることが明確化される。

【００８０】

なお、図６に示す修正後ＨＤＬ記述２Ｂでは、２ビットの信号ｄが新たに定義され〔wire［0:1］d;〕、信号ｄの２ビットのうちの一方が信号ｃに割り当てられるとともに〔assign c = d［0］;〕、信号ｄと出力ｕとが対応付けられている〔ins(.s(a),.t(b),.u(d))〕。

【００８１】

階層構造をもつＶＨＤＬによるＨＤＬ記述２Ａにおいては、上位階層のコンポーネントポートの記述と下位階層のエンティティポートの記述とが不一致である

ため、上下階層を一括して処理することが不可能になる場合がある。例えば図７に示すＨＤＬ記述２Ａでは、下位階層においてエンティティポート名Ｖが記述されているが、上位階層においてそのポート名Ｖをもつポートが空き状態であるため、そのポート名Ｖの記述が省略されている。

【００８２】

このとき、上位階層のコンポーネントポートの記述を下位階層におけるエンティティポートの記述に合わせるという修正規則５３ｃ（component port:complement to entity）を、階層情報テンプレート５３に制御情報として定義・登録しておけば、図７に示すＨＤＬ記述２Ａにおける省略記述“port(S,T:in std_logic;U:out std_logic”や“port map(S=>A,T=>B,U=>C)”が、修正後ＨＤＬ記述２Ｂにおいて、それぞれ、下位階層のエンティティポートの記述に一致させた記述“port(S,T:in std_logic;U,V:out std_logic”や“port map(S=>A,T=>B,U=>C,V=>OPEN)”に自動的に修正され、さらに、その修正箇所の行末に、コメント付与手段２０により修正コメント“//修正”が付与される。これにより、上位階層において、ポート名Ｖをもつポートの存在とそのポートが空き状態であることとが明確化される。

【００８３】

図８に示すＨＤＬ記述２Ａにおいて、Ｖerilog-ＨＤＬ（もしくはＶＨＤＬ）のポート位置対応記述やポート名対応記述はいずれも文法的に適しているが、その記述を統一する必要性がある場合などには、インスタンス部におけるポート名（端子名）を名前対応もしくは位置対応で統一的に記述するという修正規則５３ｄ（port connection:name;）を、階層情報テンプレート５３に制御情報として定義・登録する。

【００８４】

例えば、初期ＨＤＬ記述２Ａが図８に示すごとくポート位置対応記述されており、修正規則５３ｄがポート名対応記述への修正を指示していた場合、ＨＤＬ記述２Ａにおける記述“test1 test1_ins(p,q);”は、“test1 test1_ins(.a(p),.b(q));”に自動的に修正され、さらに、その修正箇所の行末に、コメント付与手段２０により修正コメント“//修正”が付与される。これにより、インスタンス

部におけるポート名（端子名）が名前対応もしくは位置対応で統一的に記述される。

【００８５】

以上のように、階層情報テンプレート５３における修正対象事項や修正規則を適当に定義することにより、ＨＤＬ記述２Ａを成す複数の階層において端子記述が不一致もしくは不統一である箇所が、一致または統一した記述になるように自動修正される。従って、従来、フロントエンド（言語処理系）ではなくバックエンド（論理合成ツールや検証ツール）でないと検出できなかった不適切な記述箇所が早期に発見されて自動修正されるので、設計工程での手戻りの発生を確実に抑止することができる。

【００８６】

〔２－５〕接続情報テンプレートを用いた修正動作について

ＨＤＬにおける信号代入記述は、ＲＴＬ（Register Transfer Level）記述では動作仕様の基本であるので、左辺と右辺とを間違う可能性は低い。しかし、構造記述部分では、単純であるが大量の記述が行なわれる場合があるので、左辺と右辺とを間違って記述することも有りうる。一般に、ＨＤＬではポート（端子）と信号代入とに関する文法が存在しており、特に、ＶＨＤＬでは、５種類のポートin，out，inout，buffer，linkageに対して以下のように使用法が規定されているので、規則を、信号代入記述の左辺と右辺とへの記述可能性として表記することが可能である。

【００８７】

in	かならず右辺でなければならない
out	かならず左辺でなければならない
inout	左辺，右辺のどちらでもよい
buffer	左辺，右辺のどちらでもよい
linkage	左辺，右辺のどちらでもよい

【００８８】

同様に、Ｖerilog-ＨＤＬでは、３種類のポートinput，output，inoutに対して以下のように使用法が規定されている。

```
input       かならず右辺でなければならない
output      かならず左辺でなければならない
inout       左辺，右辺のどちらでもよい
```

【００８９】

本実施形態のＨＤＬ自動修正装置１では、上述のような使用規則を、接続情報テンプレート５４に含まれる修正規則として適用することで、修正対象事項検出手段１７により、信号代入記述の左辺と右辺との関係が誤っている箇所が、修正対象事項に対応する箇所として検出され、さらに、修正対象事項修正手段１８により、検出された箇所における信号代入記述の左辺と右辺との関係が、上記修正規則（使用規則）に従って自動的に修正される。

【００９０】

例えば、ＶＨＤＬによる初期ＨＤＬ記述２Ａに、以下のような信号代入記述があった場合、文法違反ではないが、出力端子の信号ｂが入力端子ａに代入されるのは不適切である。

```
module test(a,b);
input a;
output b;
assign a=b;
endmodule
```

【００９１】

このとき、以下のような修正規則を接続情報テンプレート５４に定義・登録する。

```
input : right     #input端子は右辺でなければならない
output: left      #output端子は左辺でなければならない
inout: both       #inout端子は両辺どちらでもよい
```

【００９２】

これにより、ＨＤＬ記述２Ａ内に存在する信号代入記述の各々について左辺と右辺との妥当性を確認し、逆の記述があった場合、ＨＤＬ記述２Ａを自動的に修正し、修正済みのＨＤＬ記述２Ｂを生成・出力することができる。なお、その修

正箇所の行末には、コメント付与手段２０により修正コメント“//修正”が付与される。

【００９３】

修正後のＨＤＬ記述２Ｂは以下のようになる。

```
module test(a,b);
input a;
output b;
assign b=a;//修正
endmodule
```

【００９４】

以上のように接続情報テンプレート５４における修正対象事項や修正規則を適当に定義することにより、信号代入記述の左辺と右辺との関係が誤っている箇所を正しい関係に自動修正することができる。また、ＨＤＬの言語種類によって修正規則を使い分けることで、それぞれの言語仕様に応じた修正を自動的に行なうことが可能になる。

【００９５】

〔２－６〕非合成記述テンプレートを用いた修正動作について

論理合成したいＨＤＬ記述においては、論理検証時に使用した論理合成不能な波形観測用シミュレーション記述等がコメントアウトされずに残っている場合が多々ある。

【００９６】

そこで、本実施形態のＨＤＬ自動修正装置１では、非合成記述テンプレート５５に含まれる修正規則５５ａ（図９参照）において、論理合成不能な波形観測用シミュレーション記述文を全て列挙するとともに、その記述文を削除するか、もしくは、その記述文を論理合成ツールに無視させるためのディレクティブを追加・記入するかを指定する。なお、ディレクティブは、該当記述文の前後を囲むように記述され、このようにディレクティブで囲まれた記述文は、論理合成ツールによって無視され論理合成対象外の記述として取り扱われる。なお、該当記述文を削除するかディレクティブで囲むかは設計者によって適宜選択される。

【００９７】

例えば図９では、非合成記述テンプレート５５に含まれる修正規則５５ａにおいて、論理合成不能な波形観測用シミュレーション記述文として“initial文”と“$monitor()”とが列挙され、これらの記述文を“synthesis on/off directive”で囲むように指定されている。

【００９８】

図９に具体的に示すＨＤＬ記述２Ａにおいて、“initial o = 1' b0;”と“$monitor(o);”とが論理合成不能な波形観測用シミュレーション記述文であり、これらの記述文が修正対象事項検出手段１７により検出される。そして、修正対象事項修正手段１８により、その記述文の前後に“synthesis on/off directive”が自動的に追加・記入され（図９の修正後ＨＤＬ記述２Ｂ参照）、さらに、その追加・記入箇所（修正箇所）の行末に、コメント付与手段２０により修正コメント“//修正”が付与される。

【００９９】

これにより、ＨＤＬ記述２Ａにおいて、例えば、論理検証時に使用した論理合成不能な波形観測用シミュレーション記述文“initial o = 1' b0;”や“$monitor(o);”がコメントアウトされずに残っていたとしても、このような記述文が論理合成ツールにおいて問題（エラー）を引き起こすことがなくなる。従って、上述のような記述文を設計者が手作業により削除するといった作業が不要になり、設計者にかかる負担を大幅に削減することができる。

【０１００】

〔３〕本発明の一実施形態の効果の説明

このように、本発明の一実施形態としてのＨＤＬ自動修正装置１によれば、初期ＨＤＬ記述２Ａに存在する不適切な記述を検査することが可能となり、さらに修正を施されたＨＤＬ記述２Ｂが生成されるので、高品質のＨＤＬ記述２Ｂを得ることができる。

【０１０１】

特に、本実施形態では、重度の意味論的文法違反が自動修正され且つその修正箇所が明確化されるので、設計者にかかる負担が大幅に軽減されるとともに、高

品質のＨＤＬ記述２Ｂを得ることができる。また、文法違反ではないが回路設計上考慮すべき箇所（修正対象事項に対応する箇所）が適切な記述に自動修正され且つその修正箇所が明確化されるので、設計者にかかる負担が大幅に軽減されるとともに、より高品質のＨＤＬ記述２Ｂを得ることができる。

【０１０２】

さらに、テンプレート５１～５５における修正対象事項や修正規則を適当に定義することにより、回路設計上考慮すべき箇所（不適切な記述）や設計者の単純ミスを早期に発見して確実に自動修正することができるので、設計工程での手戻りの発生を確実に抑止することが可能になる。

不適切な記述は多くの場合、設計者の設計意図とは異なる回路の機能や構造を表現しているので、後続の論理検証や回路実装の工程になって誤りが発見され修正作業が発生するが、本実施形態のＨＤＬ自動修正装置１を用いて、不適切な記述をＨＤＬ記述の作成時に修正することにより、無駄な修正作業を発生させずに済む。

【０１０３】

一方、修正に関するコメント（注意書き）を修正箇所に付与することにより、修正箇所が明確化され、設計者は、ＨＤＬ記述２Ｂのどこをどのように修正したかを容易に視認することができる。従って、設計者は、自動修正結果が意図するものかどうか、即ち適正な修正が行なわれたか否かの確認作業を容易かつ確実に行なえ、設計者にかかる負担が大幅に削減される。

【０１０４】

また、修正後ＨＤＬ記述２Ｂに付与された修正コメントにより、ＨＤＬによって回路設計情報を記述する際に、どのような箇所において、修正が必要な記述をする可能性があるかを設計者に把握させることができるので、設計者に対する教育的効果も得られる。

【０１０５】

〔４〕その他

なお、本発明は上述した実施形態に限定されるものではなく、本発明の趣旨を逸脱しない範囲で種々変形して実施することができる。

例えば、上述した実施形態では、ＨＤＬがＶＨＤＬやVerilog-ＨＤＬである場合について説明したが、本発明は、これらのVerilog-ＨＤＬやＶＨＤＬ以外の言語にも適用可能であり、その場合も上述と同様の作用効果が得られ、回路設計分野やソフトウェア開発分野における効率化・負担削減に大きく寄与するものである。

【０１０６】

〔５〕付記

（付記１）　ハードウェア記述言語（以下、ＨＤＬという）によって記述された回路設計情報（以下、ＨＤＬ記述という）を自動的に修正する装置であって、

修正対象のＨＤＬ記述の字句解析を行なうＨＤＬ字句解析手段と、

該ＨＤＬ字句解析手段による解析結果に基づいて該ＨＤＬ記述の構文解析を行ない、該ＨＤＬ記述を解析木形式の記述に変換するＨＤＬ構文解析手段と、

該ＨＤＬ構文解析手段による解析結果に基づいて該ＨＤＬ記述の意味論解析を行ない、該ＨＤＬ記述における代入文の右辺と左辺の変数の型が不一致になっている部分を意味論的文法違反箇所として検出する意味論的文法違反検出手段と、

前記代入文の右辺の変数の型を左辺の変数の型に変換する型変換関数を、型変換規則として定義する型変換テンプレートと、

該意味論的文法違反検出手段によって該意味論的文法違反箇所とみなされた代入文の右辺に対して、該型変換テンプレートに定義された該型変換関数を適用することにより、該意味論的文法違反箇所を正しい記述に修正する意味論的文法違反修正手段と、

該意味論的文法違反修正手段による修正を施された該ＨＤＬ記述の逆構文解析を行ない、該ＨＤＬ記述を前記解析木形式の記述から通常の記述形式に変換するＨＤＬ逆構文解析手段と、

該意味論的文法違反修正手段によって修正を施された箇所に、該修正に関するコメントを付与するコメント付与手段とをそなえて構成されたことを特徴とする、ＨＤＬ自動修正装置。

【０１０７】

（付記２）　文法違反ではないが回路設計上考慮すべき修正対象事項、および、該修正対象事項に対応する箇所の修正規則を定義する制御情報テンプレートと、

該ＨＤＬ構文解析手段による解析結果に基づき、該ＨＤＬ記述において、該修正対象事項に対応する箇所を検出する修正対象事項検出手段と、

該修正対象事項検出手段によって検出された箇所を、該制御情報テンプレートに定義された該修正規則に従って修正する修正対象事項修正手段とをさらにそなえ、

該ＨＤＬ逆構文解析手段が、該意味論的文法違反修正手段および該修正対象事項修正手段による修正を施された該ＨＤＬ記述の逆構文解析を行なうとともに、

該コメント付与手段が、該意味論的文法違反修正手段および該修正対象事項修正手段によって修正を施された箇所に、該コメントを付与することを特徴とする、付記１記載のＨＤＬ自動修正装置。

【０１０８】

（付記３）　ハードウェア記述言語（以下、ＨＤＬという）によって記述された回路設計情報（以下、ＨＤＬ記述という）を自動的に修正する装置であって、

修正対象のＨＤＬ記述の字句解析を行なうＨＤＬ字句解析手段と、

該ＨＤＬ字句解析手段による解析結果に基づいて該ＨＤＬ記述の構文解析を行ない、該ＨＤＬ記述を解析木形式の記述に変換するＨＤＬ構文解析手段と、

文法違反ではないが回路設計上考慮すべき修正対象事項、および、該修正対象事項に対応する箇所の修正規則を定義する制御情報テンプレートと、

該ＨＤＬ構文解析手段による解析結果に基づき、該ＨＤＬ記述において、該修正対象事項に対応する箇所を検出する修正対象事項検出手段と、

該修正対象事項検出手段によって検出された箇所を、該制御情報テンプレートに定義された該修正規則に従って修正する修正対象事項修正手段と、

該修正対象事項修正手段による修正を施された該ＨＤＬ記述の逆構文解析を行ない、該ＨＤＬ記述を前記解析木形式の記述から通常の記述形式に変換するＨＤＬ逆構文解析手段と、

該修正対象事項修正手段によって修正を施された箇所に、該修正に関するコメントを付与するコメント付与手段とをそなえて構成されたことを特徴とする、ＨＤＬ自動修正装置。

【０１０９】

（付記４）　該修正対象事項検出手段が、処理中の該ＨＤＬを異なる種類の他のＨＤＬに変換した場合に該他のＨＤＬの言語規約を満たさない箇所を、該修正対象事項に対応する箇所として検出するように、該制御情報テンプレートにおける該修正対象事項が定義されるとともに、

該修正対象事項修正手段が、該修正対象事項検出手段によって検出された箇所を、該他のＨＤＬの言語規約を満たす記述に変換・修正するように、該制御情報テンプレートにおける該修正規則が定義されることを特徴とする、付記２または付記３に記載のＨＤＬ自動修正装置。

【０１１０】

（付記５）　処理中の該ＨＤＬがアルファベットの大文字と小文字を区別して使用するものである場合、処理中の該ＨＤＬを、大文字と小文字を区別しないで使用する他のＨＤＬに変換する可能性を考慮し、

該修正対象事項検出手段が、大文字と小文字を区別して記述された同一綴りの一対の文字列のうちの一方を、該修正対象事項に対応する箇所として検出するように、該制御情報テンプレートにおける該修正対象事項が定義されるとともに、

該修正対象事項修正手段が、該ＨＤＬ記述中に存在する文字列の綴りと重複しない綴りの新しい文字列を生成してから、該修正対象事項検出手段によって検出された該一方の文字列を該新しい文字列に置換するように、該制御情報テンプレートにおける該修正規則が定義されることを特徴とする、付記４記載のＨＤＬ自動修正装置。

【０１１１】

（付記６）　処理中の該ＨＤＬがアルファベットの大文字と小文字を区別しないで使用するものである場合、処理中の該ＨＤＬを、大文字と小文字を区別して使用する他のＨＤＬに変換する可能性を考慮し、

該修正対象事項検出手段が、該ＨＤＬ記述の文字列における大文字および小文

字のうちの一方を、該修正対象事項に対応する箇所として検出するように、該制御情報テンプレートにおける該修正対象事項が定義されるとともに、

該修正対象事項修正手段が、該修正対象事項検出手段によって検出された大文字または小文字をそれぞれ小文字または大文字に変換するように、該制御情報テンプレートにおける該修正規則が定義されることを特徴とする、付記４記載のＨＤＬ自動修正装置。

【０１１２】

（付記７）　該修正対象事項検出手段が、所定の使用禁止文字を含む文字列を、該修正対象事項に対応する箇所として検出するように、該制御情報テンプレートにおける該修正対象事項が定義されるとともに、

該修正対象事項修正手段が、該ＨＤＬ記述に存在する文字列と重複せず且つ該所定の使用禁止文字を含まない新しい文字列を生成してから、該修正対象事項検出手段によって検出された該使用禁止文字を含む文字列を該新しい文字列に置換するように、該制御情報テンプレートにおける該修正規則が定義されることを特徴とする、付記２～付記６のいずれか一つに記載のＨＤＬ自動修正装置。

【０１１３】

（付記８）　該修正対象事項検出手段が、該ＨＤＬ記述を成す複数の階層において端子記述が不一致もしくは不統一である箇所を、該修正対象事項に対応する箇所として検出するように、該制御情報テンプレートにおける該修正対象事項が定義されるとともに、

該修正対象事項修正手段が、該修正対象事項検出手段によって検出された箇所における該端子記述を、該複数の階層の全てにおいて一致または統一した記述に修正するように、該制御情報テンプレートにおける該修正規則が定義されることを特徴とする、付記２～付記７のいずれか一つに記載のＨＤＬ自動修正装置。

【０１１４】

（付記９）　該修正対象事項検出手段が、信号代入記述の左辺と右辺との関係が誤っている箇所を、該修正対象事項に対応する箇所として検出するように、該制御情報テンプレートにおける該修正対象事項が定義されるとともに、

該修正対象事項修正手段が、該修正対象事項検出手段によって検出された箇所

における該信号代入記述の左辺と右辺との関係を修正するように、該制御情報テンプレートにおける該修正規則が定義されることを特徴とする、付記２～付記８のいずれか一つに記載のＨＤＬ自動修正装置。

【０１１５】

（付記１０）　該修正対象事項検出手段が、論理合成ツールが合成することのできない箇所を、該修正対象事項に対応する箇所として検出するように、該制御情報テンプレートにおける該修正対象事項が定義されるとともに、

該修正対象事項修正手段が、該修正対象事項検出手段によって検出された箇所を削除するように、該制御情報テンプレートにおける該修正規則が定義されることを特徴とする、付記２～付記９のいずれか一つに記載のＨＤＬ自動修正装置。

【０１１６】

（付記１１）　該修正対象事項検出手段が、論理合成ツールが合成することのできない箇所を、該修正対象事項に対応する箇所として検出するように、該制御情報テンプレートにおける該修正対象事項が定義されるとともに、

該修正対象事項修正手段が、該修正対象事項検出手段によって検出された箇所に、当該箇所における記述を該論理合成ツールに無視させるためのディレクティブを追加・記入するように、該制御情報テンプレートにおける該修正規則が定義されることを特徴とする、付記２～付記９のいずれか一つに記載のＨＤＬ自動修正装置。

【０１１７】

（付記１２）　ハードウェア記述言語（以下、ＨＤＬという）によって記述された回路設計情報（以下、ＨＤＬ記述という）を、コンピュータに自動修正させるためのＨＤＬ自動修正プログラムであって、

修正対象のＨＤＬ記述の字句解析を行なうＨＤＬ字句解析手段、

該ＨＤＬ字句解析手段による解析結果に基づいて該ＨＤＬ記述の構文解析を行ない、該ＨＤＬ記述を解析木形式の記述に変換するＨＤＬ構文解析手段、

該ＨＤＬ構文解析手段による解析結果に基づいて該ＨＤＬ記述の意味論解析を行ない、該ＨＤＬ記述における代入文の右辺と左辺の変数の型が不一致になっている部分を意味論的文法違反箇所として検出する意味論的文法違反検出手段、

該意味論的文法違反検出手段によって該意味論的文法違反箇所とみなされた代入文の右辺に対して、前記代入文の右辺の変数の型を左辺の変数の型に変換する型変換関数を適用することにより、該意味論的文法違反箇所を正しい記述に修正する意味論的文法違反修正手段、

該意味論的文法違反修正手段による修正を施された該ＨＤＬ記述の逆構文解析を行ない、該ＨＤＬ記述を前記解析木形式の記述から通常の記述形式に変換するＨＤＬ逆構文解析手段、および、

該意味論的文法違反修正手段によって修正を施された箇所に、該修正に関するコメントを付与するコメント付与手段として、該コンピュータを機能させることを特徴とする、ＨＤＬ自動修正プログラム。

【０１１８】

（付記１３）　該ＨＤＬ構文解析手段による解析結果に基づき、該ＨＤＬ記述において、文法違反ではないが回路設計上考慮すべき修正対象事項に対応する箇所を検出する修正対象事項検出手段、および、

該修正対象事項検出手段によって検出された箇所を、予め定義された該修正対象事項に対応する箇所の修正規則に従って修正する修正対象事項修正手段として、該コンピュータをさらに機能させ、

該ＨＤＬ逆構文解析手段に、該意味論的文法違反修正手段および該修正対象事項修正手段による修正を施された該ＨＤＬ記述の逆構文解析を行なわせるとともに、

該コメント付与手段に、該意味論的文法違反修正手段および該修正対象事項修正手段によって修正を施された箇所に、該コメントを付与させることを特徴とする、付記１２記載のＨＤＬ自動修正プログラム。

【０１１９】

（付記１４）　ハードウェア記述言語（以下、ＨＤＬという）によって記述された回路設計情報（以下、ＨＤＬ記述という）を、コンピュータに自動修正させるためのＨＤＬ自動修正プログラムであって、

修正対象のＨＤＬ記述の字句解析を行なうＨＤＬ字句解析手段、

該ＨＤＬ字句解析手段による解析結果に基づいて該ＨＤＬ記述の構文解析を行

ない、該ＨＤＬ記述を解析木形式の記述に変換するＨＤＬ構文解析手段、

該ＨＤＬ構文解析手段による解析結果に基づき、該ＨＤＬ記述において、文法違反ではないが回路設計上考慮すべき修正対象事項に対応する箇所を検出する修正対象事項検出手段、

該修正対象事項検出手段によって検出された箇所を、予め定義された該修正対象事項に対応する箇所の修正規則に従って修正する修正対象事項修正手段、

該修正対象事項修正手段による修正を施された該ＨＤＬ記述の逆構文解析を行ない、該ＨＤＬ記述を前記解析木形式の記述から通常の記述形式に変換するＨＤＬ逆構文解析手段、および、

該修正対象事項修正手段によって修正を施された箇所に、該修正に関するコメントを付与するコメント付与手段として、該コンピュータを機能させることを特徴とする、ＨＤＬ自動修正プログラム。

【０１２０】

（付記１５）　ハードウェア記述言語（以下、ＨＤＬという）によって記述された回路設計情報（以下、ＨＤＬ記述という）を、コンピュータに自動修正させるためのＨＤＬ自動修正プログラムを記録したコンピュータ読取可能な記録媒体であって、

該ＨＤＬ自動修正プログラムが、

修正対象のＨＤＬ記述の字句解析を行なうＨＤＬ字句解析手段、

該ＨＤＬ字句解析手段による解析結果に基づいて該ＨＤＬ記述の構文解析を行ない、該ＨＤＬ記述を解析木形式の記述に変換するＨＤＬ構文解析手段、

該ＨＤＬ構文解析手段による解析結果に基づいて該ＨＤＬ記述の意味論解析を行ない、該ＨＤＬ記述における代入文の右辺と左辺の変数の型が不一致になっている部分を意味論的文法違反箇所として検出する意味論的文法違反検出手段、

を、型変換規則として定義する型変換テンプレート、

該意味論的文法違反検出手段によって該意味論的文法違反箇所とみなされた代入文の右辺に対して、前記代入文の右辺の変数の型を左辺の変数の型に変換する型変換関数を適用することにより、該意味論的文法違反箇所を正しい記述に修正する意味論的文法違反修正手段、

該意味論的文法違反修正手段による修正を施された該ＨＤＬ記述の逆構文解析を行ない、該ＨＤＬ記述を前記解析木形式の記述から通常の記述形式に変換するＨＤＬ逆構文解析手段、および、

該意味論的文法違反修正手段によって修正を施された箇所に、該修正に関するコメントを付与するコメント付与手段として、該コンピュータを機能させることを特徴とする、ＨＤＬ自動修正プログラムを記録したコンピュータ読取可能な記録媒体。

【０１２１】

（付記１６）　該ＨＤＬ自動修正プログラムが、

該ＨＤＬ構文解析手段による解析結果に基づき、該ＨＤＬ記述において、文法違反ではないが回路設計上考慮すべき修正対象事項に対応する箇所を検出する修正対象事項検出手段、および、

該修正対象事項検出手段によって検出された箇所を、予め定義された該修正対象事項に対応する箇所の修正規則に従って修正する修正対象事項修正手段として、該コンピュータをさらに機能させ、

該ＨＤＬ逆構文解析手段に、該意味論的文法違反修正手段および該修正対象事項修正手段による修正を施された該ＨＤＬ記述の逆構文解析を行なわせるとともに、

該コメント付与手段に、該意味論的文法違反修正手段および該修正対象事項修正手段によって修正を施された箇所に、該コメントを付与させることを特徴とする、付記１５記載のＨＤＬ自動修正プログラムを記録したコンピュータ読取可能な記録媒体。

【０１２２】

（付記１７）　ハードウェア記述言語（以下、ＨＤＬという）によって記述された回路設計情報（以下、ＨＤＬ記述という）を、コンピュータに自動修正させるためのＨＤＬ自動修正プログラムを記録したコンピュータ読取可能な記録媒体であって、

該ＨＤＬ自動修正プログラムが、

修正対象のＨＤＬ記述の字句解析を行なうＨＤＬ字句解析手段、

該ＨＤＬ字句解析手段による解析結果に基づいて該ＨＤＬ記述の構文解析を行ない、該ＨＤＬ記述を解析木形式の記述に変換するＨＤＬ構文解析手段、

該ＨＤＬ構文解析手段による解析結果に基づき、該ＨＤＬ記述において、文法違反ではないが回路設計上考慮すべき修正対象事項に対応する箇所を検出する修正対象事項検出手段、

該修正対象事項検出手段によって検出された箇所を、予め定義された該修正対象事項に対応する箇所の修正規則に従って修正する修正対象事項修正手段、

該修正対象事項修正手段による修正を施された該ＨＤＬ記述の逆構文解析を行ない、該ＨＤＬ記述を前記解析木形式の記述から通常の記述形式に変換するＨＤＬ逆構文解析手段、および、

該修正対象事項修正手段によって修正を施された箇所に、該修正に関するコメントを付与するコメント付与手段として、該コンピュータを機能させることを特徴とする、ＨＤＬ自動修正プログラムを記録したコンピュータ読取可能な記録媒体。

【０１２３】

【発明の効果】

以上詳述したように、本発明のＨＤＬ自動修正装置（請求項１～３）およびＨＤＬ自動修正プログラム（請求項４）並びに同プログラムを記録したコンピュータ読取可能な記録媒体（請求項５）によれば、以下のような効果ないし利点を得ることができる。

（１）重度の意味論的文法違反が自動修正され且つその修正箇所が明確化されるので、設計者にかかる負担が大幅に軽減されるとともに、高品質のＨＤＬ記述を得ることができる（請求項１，４，５）。

【０１２４】

（２）文法違反ではないが回路設計上考慮すべき箇所が適切な記述に自動修正され且つその修正箇所が明確化されるので、設計者にかかる負担が大幅に軽減されるとともに、高品質のＨＤＬ記述を得ることができる（請求項２，３）。

（３）制御情報テンプレートにおける修正対象事項や修正規則を適当に定義することにより、回路設計上考慮すべき箇所（不適切な記述）や設計者の単純ミス

を早期に発見して確実に自動修正することができるので、設計工程での手戻りの発生を確実に抑止することが可能になる（請求項２，３）。

【０１２５】

．（４）修正に関するコメントを修正箇所に付与することにより、設計者は、ＨＤＬ記述のどこをどのように修正したかを容易に視認することができる。従って、設計者は、自動修正結果が意図するものかどうか、即ち適正な修正が行なわれたか否かの確認作業を容易かつ確実に行なえ、設計者にかかる負担が大幅に削減される（請求項１～５）。

【０１２６】

（５）修正に関するコメントを修正箇所に付与することにより、ＨＤＬによって回路設計情報を記述する際に、どのような箇所において、修正が必要な記述をする可能性があるかを設計者に把握させることができ、設計者に対する教育的効果も得られる（請求項１～５）。

【０１２７】

（６）制御情報テンプレートにおける修正対象事項や修正規則を適当に定義することにより、ＨＤＬ記述を処理中のＨＤＬから他のＨＤＬに変換する可能性がある場合、他のＨＤＬに変換した時に回路設計上の問題が生じないように、予め他のＨＤＬの言語規約も満たすようにＨＤＬ記述を自動修正することができる。従って、設計者に負担をかけることなく、複数種類のＨＤＬにおいて回路設計上の問題を生じさせることのないＨＤＬ記述を得ることができる。

【０１２８】

（７）制御情報テンプレートにおける修正対象事項や修正規則を適当に定義することにより、使用禁止文字を含む文字列が、ＨＤＬ記述に存在する文字列と重複せず且つ使用禁止文字を含まない新しい文字列に自動修正される。従って、例えば複数の設計者によりＨＤＬ記述を作成した場合に命名規則に違反する箇所が多数出現したとしても、設計者に負担をかけることなく、その違反箇所の名前（文字列）を容易かつ確実に修正して命名規則に従ったＨＤＬ記述を得ることができる。

【０１２９】

（８）制御情報テンプレートにおける修正対象事項や修正規則を適当に定義することにより、ＨＤＬ記述を成す複数の階層において端子記述が不一致もしくは不統一である箇所が、一致または統一した記述になるように自動修正される。従って、従来、フロントエンド（言語処理系）ではなくバックエンド（論理合成ツールや検証ツール）でないと検出できなかった不適切な記述箇所が早期に発見されて自動修正されるので、設計工程での手戻りの発生を確実に抑止することができる。

【０１３０】

（９）制御情報テンプレートにおける修正対象事項や修正規則を適当に定義することにより、信号代入記述の左辺と右辺との関係が誤っている箇所を正しい関係に自動修正することができる。

【０１３１】

（１０）制御情報テンプレートにおける修正対象事項や修正規則を適当に定義することにより、論理合成ツールが合成することのできない箇所を自動的に削除したり、その箇所における記述を論理合成ツールに無視させるためのディレクティブを自動的に追加・記入したりすることができる。これにより、ＨＤＬ記述において、例えば、論理検証時に使用した論理合成不能な波形観測用シミュレーション記述等がコメントアウトされずに残っていたとしても、このような記述が論理合成ツールにおいて問題を引き起こすことがなくなる。従って、上述のような記述を設計者が手作業により削除するといった作業が不要になり、設計者にかかる負担を大幅に削減することができる。

【図面の簡単な説明】

【図１】

本発明の一実施形態としてのＨＤＬ自動修正装置の構成を示すブロック図である。

【図２】

本実施形態の動作を説明するための図である。

【図３】

本実施形態の動作を説明するための図である。

【図４】

本実施形態の動作を説明するための図である。

【図５】

本実施形態の動作を説明するための図である。

【図６】

本実施形態の動作を説明するための図である。

【図７】

本実施形態の動作を説明するための図である。

【図８】

本実施形態の動作を説明するための図である。

【図９】

本実施形態の動作を説明するための図である。

【符号の説明】

１　ＨＤＬ自動修正装置

２Ａ　修正対象のＨＤＬ記述（初期ＨＤＬ記述）

２Ｂ　修正後のＨＤＬ記述

１１　ＨＤＬ字句解析手段

１２　ＨＤＬ構文解析手段

１３　構文的文法違反検出手段

１４　構文的文法違反修正手段

１５　意味論的文法違反検出手段

１６　意味論的文法違反修正手段

１７　修正対象事項検出手段

１８　修正対象事項修正手段

１９　ＨＤＬ逆構文解析手段

２０　コメント付与手段

２１　トークンデータベース

２２～２４　ＨＤＬデータベース

３０　文法解析テンプレート

４０　型変換テンプレート

５１　言語変換規則テンプレート（制御情報テンプレート）

５１ａ　予約語テンプレート

５１ｂ　名前テンプレート

５１ｃ　名前生成ルール

５１ｄ　大文字小文字ルール

５２　使用禁止文字情報テンプレート（制御情報テンプレート）

５２ａ　使用禁止文字テンプレート

５２ｂ　名前生成ルール

５３　階層情報テンプレート（制御情報テンプレート）

５３ａ～５３ｄ　修正規則

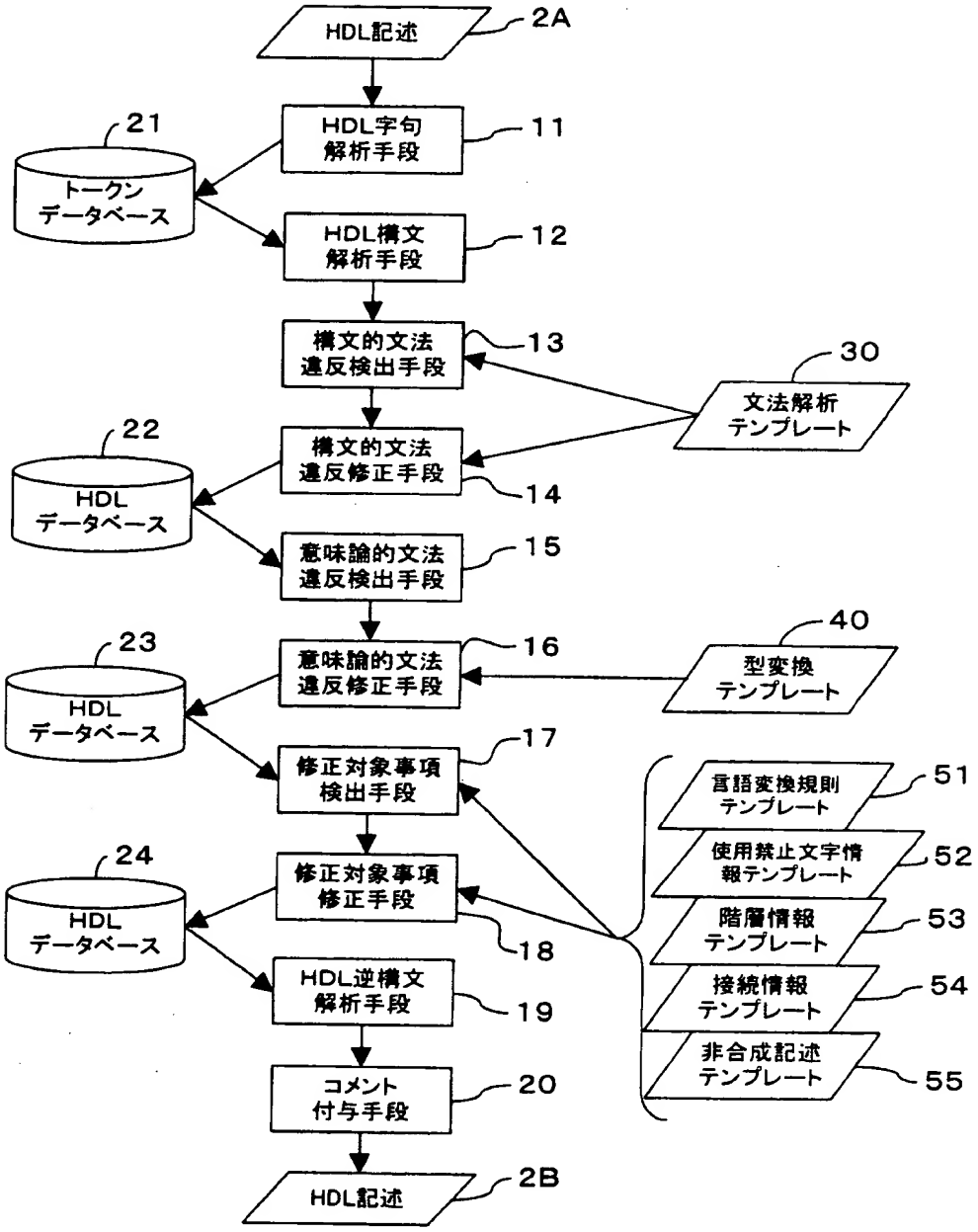
５４　接続情報テンプレート（制御情報テンプレート）

５５　非合成記述テンプレート（制御情報テンプレート）

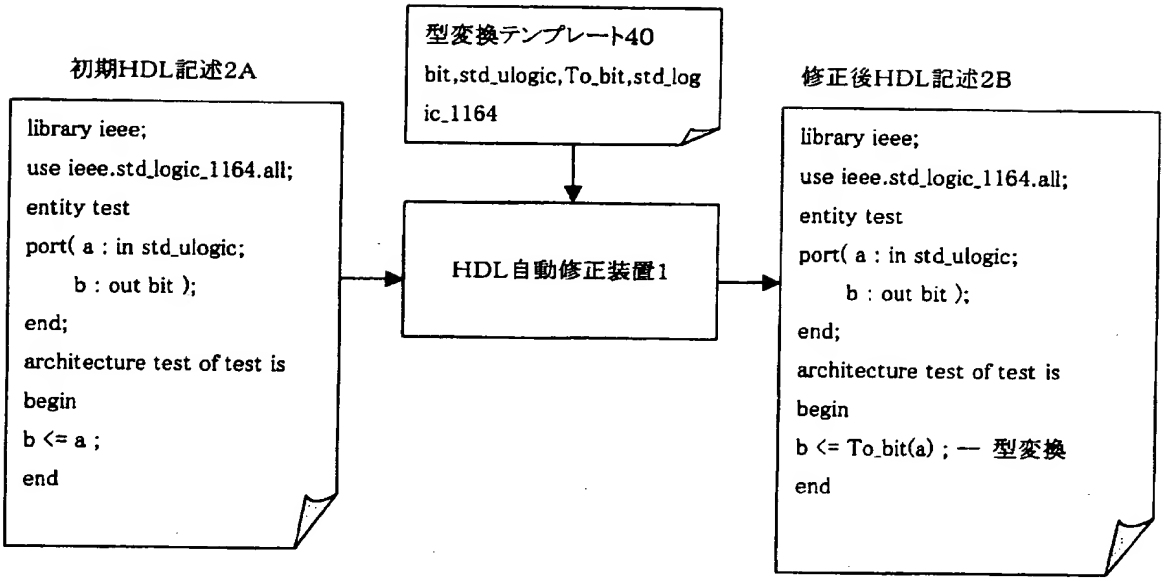
５５ａ　修正規則

【書類名】　　図面

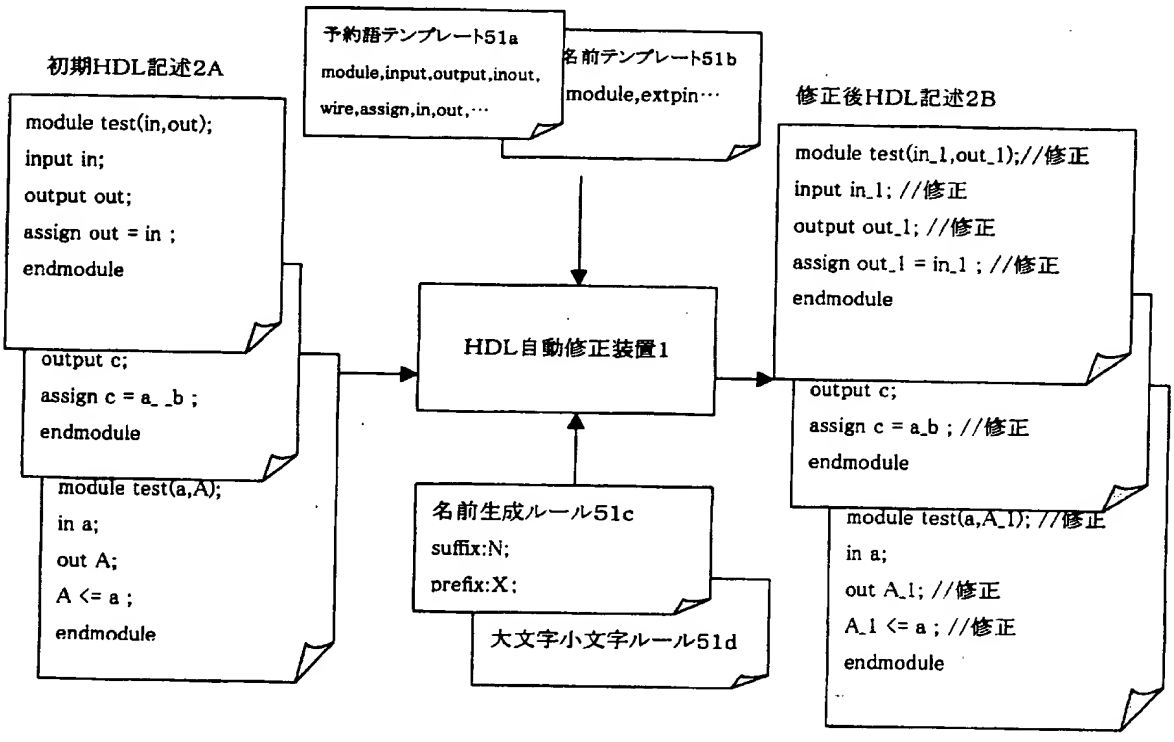
【図１】


HDL記述
2A
HDL字句
解析手段
11
21
トークン
データベース
HDL構文
解析手段
12
構文的文法
違反検出手段
13
30
文法解析
テンプレート
22
構文的文法
違反修正手段
14
HDL
データベース
意味論的文法
違反検出手段
15
40
23
意味論的文法
違反修正手段
16
型変換
テンプレート
HDL
データベース
修正対象事項
検出手段
17
言語変換規則
テンプレート
51
使用禁止文字情
報テンプレート
52
24
修正対象事項
修正手段
階層情報
テンプレート
53
HDL
データベース
18
接続情報
テンプレート
54
HDL逆構文
解析手段
19
非合成記述
テンプレート
55
コメント
付与手段
20
HDL記述
2B

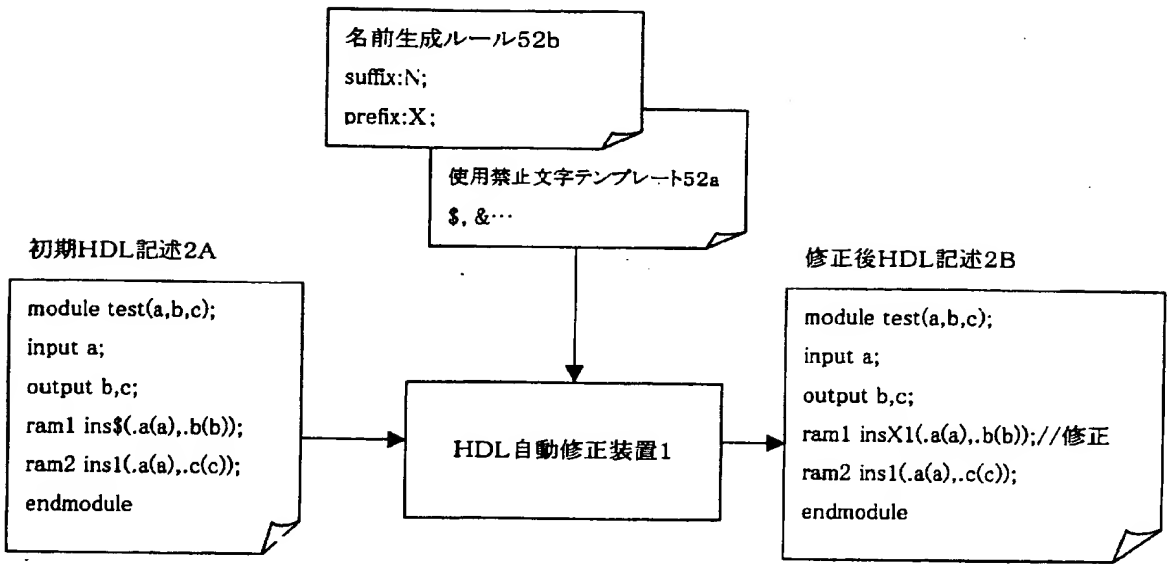
【図２】


初期HDL記述2A
library ieee;
use ieee.std_logic_1164.all;
entity test
port(a : in std_ulogic;
b : out bit);
end;
architecture test of test is
begin
b <= a ;
end
型変換テンプレート40
bit,std_ulogic,To_bit,std_logic_1164
HDL自動修正装置1
修正後HDL記述2B
library ieee;
use ieee.std_logic_1164.all;
entity test
port(a : in std_ulogic;
b : out bit);
end;
architecture test of test is
begin
b <= To_bit(a) ; — 型変換
end

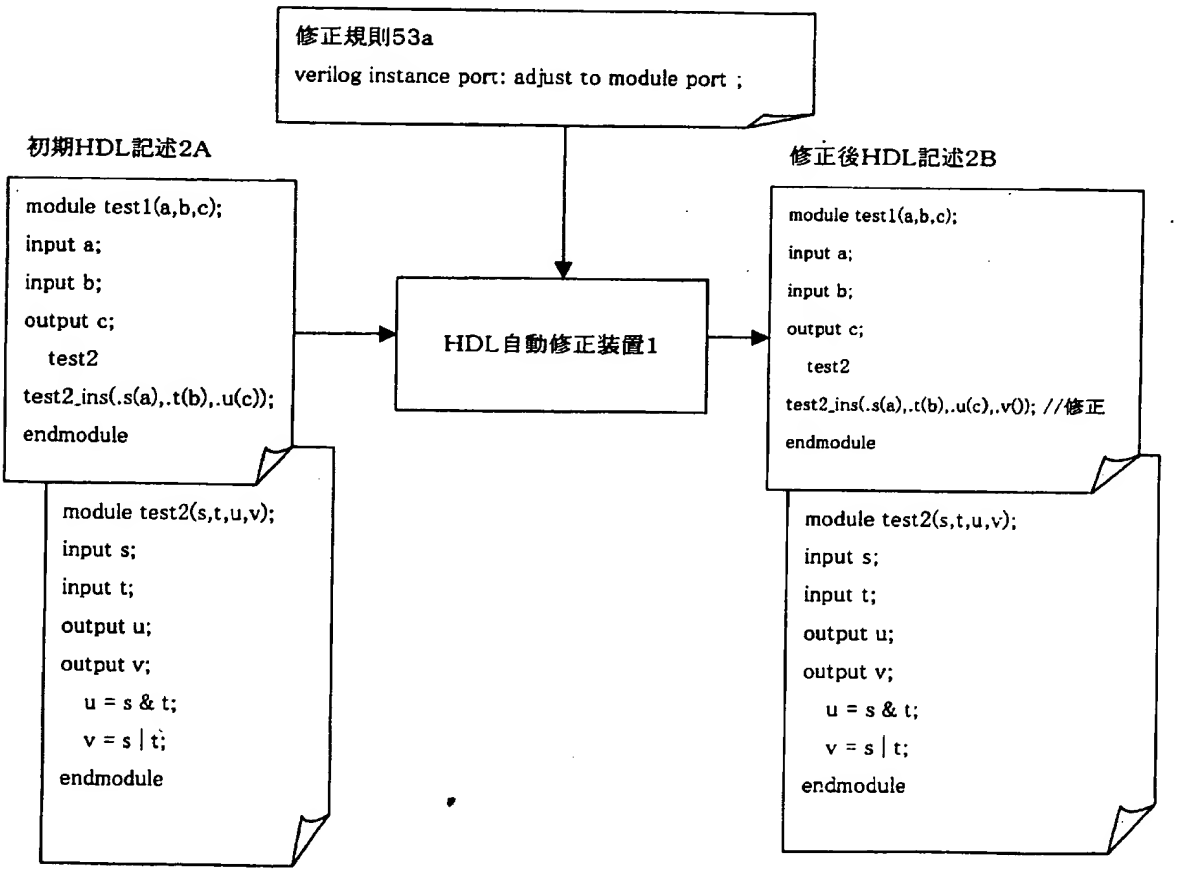
【図３】


初期HDL記述2A
module test(in,out);
input in;
output out;
assign out = in ;
endmodule
output c;
assign c = a_ _b ;
endmodule
module test(a,A);
in a;
out A;
A <= a ;
endmodule
予約語テンプレート51a
module,input,output,inout,
wire,assign,in,out,…
名前テンプレート51b
module,extpin…
HDL自動修正装置1
名前生成ルール51c
suffix:N;
prefix:X;
大文字小文字ルール51d
修正後HDL記述2B
module test(in_1,out_1);//修正
input in_1; //修正
output out_1; //修正
assign out_1 = in_1 ; //修正
endmodule
output c;
assign c = a_b ; //修正
endmodule
module test(a,A_1); //修正
in a;
out A_1; //修正
A_1 <= a ; //修正
endmodule

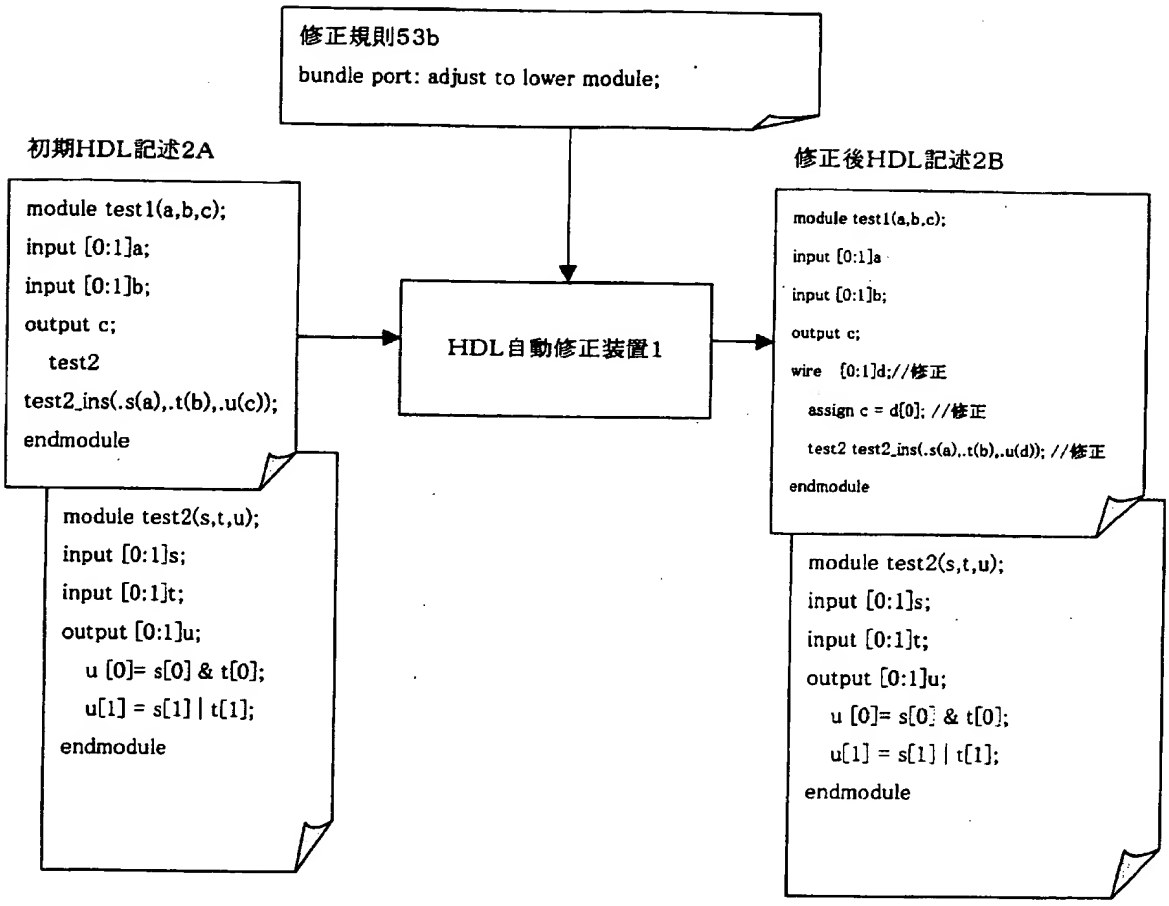
【図４】


名前生成ルール52b
suffix:N;
prefix:X;
使用禁止文字テンプレート52a
$, &…
初期HDL記述2A
module test(a,b,c);
input a;
output b,c;
ram1 ins$(.a(a),.b(b));
ram2 ins1(.a(a),.c(c));
endmodule
HDL自動修正装置1
修正後HDL記述2B
module test(a,b,c);
input a;
output b,c;
ram1 insX1(.a(a),.b(b));//修正
ram2 ins1(.a(a),.c(c));
endmodule

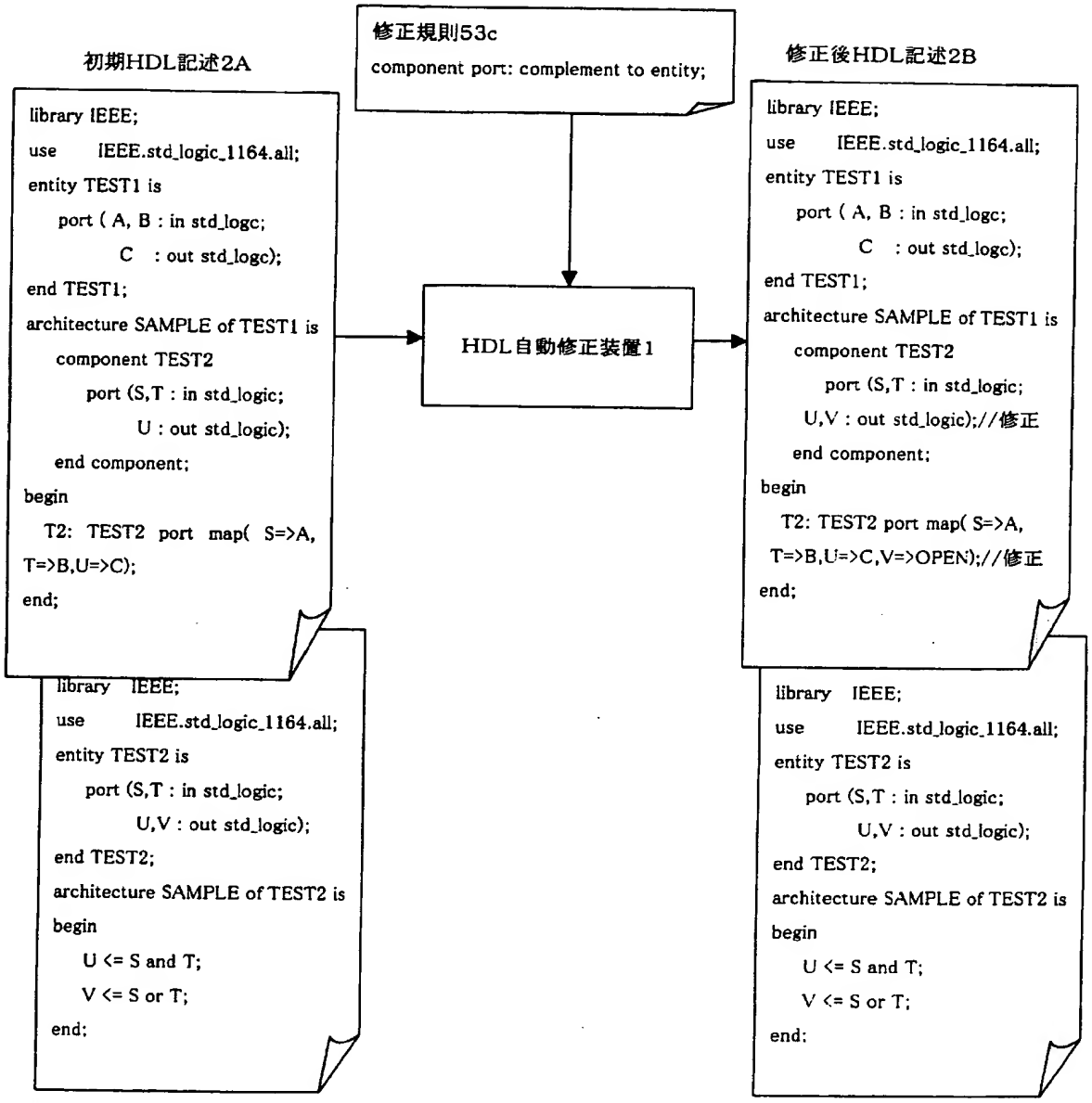
【図５】


修正規則53a
verilog instance port: adjust to module port ;
初期HDL記述2A
module test1(a,b,c);
input a;
input b;
output c;
test2
test2_ins(.s(a),.t(b),.u(c));
endmodule
module test2(s,t,u,v);
input s;
input t;
output u;
output v;
u = s & t;
v = s | t;
endmodule
HDL自動修正装置1
修正後HDL記述2B
module test1(a,b,c);
input a;
input b;
output c;
test2
test2_ins(.s(a),.t(b),.u(c),.v()); //修正
endmodule
module test2(s,t,u,v);
input s;
input t;
output u;
output v;
u = s & t;
v = s | t;
endmodule

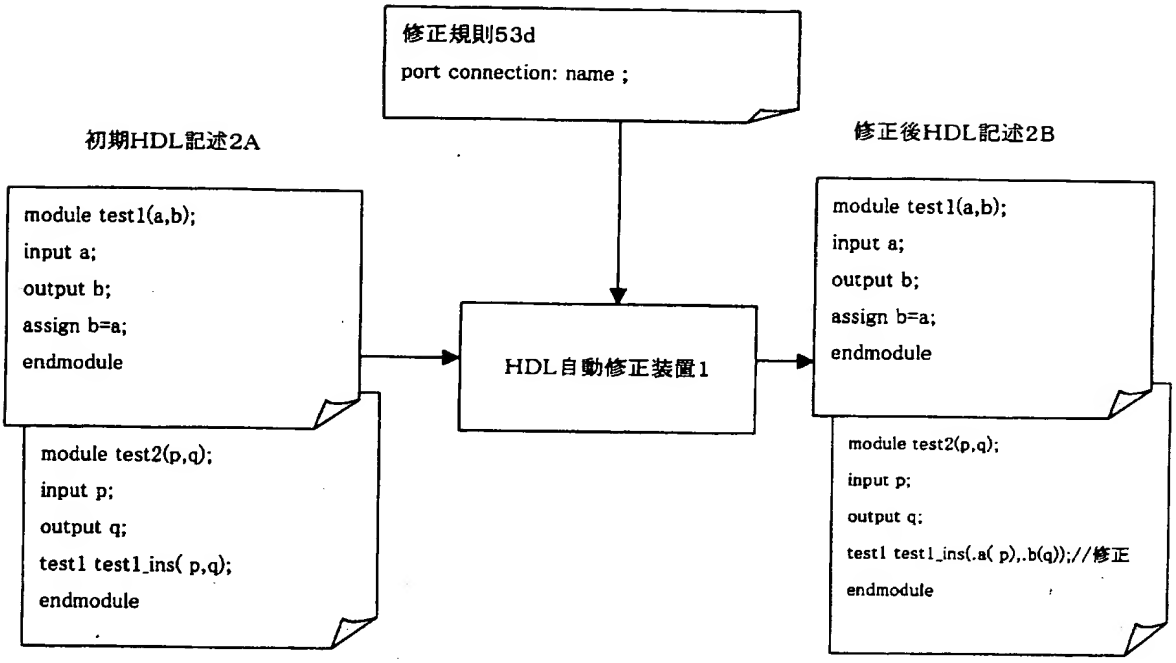
【図６】


修正規則53b
bundle port: adjust to lower module;
初期HDL記述2A
module test1(a,b,c);
input [0:1]a;
input [0:1]b;
output c;
test2
test2_ins(.s(a),.t(b),.u(c));
endmodule
module test2(s,t,u);
input [0:1]s;
input [0:1]t;
output [0:1]u;
u [0]= s[0] & t[0];
u[1] = s[1] | t[1];
endmodule
HDL自動修正装置1
修正後HDL記述2B
module test1(a,b,c);
input [0:1]a
input [0:1]b;
output c;
wire [0:1]d;//修正
assign c = d[0]; //修正
test2 test2_ins(.s(a),.t(b),.u(d)); //修正
endmodule
module test2(s,t,u);
input [0:1]s;
input [0:1]t;
output [0:1]u;
u [0]= s[0] & t[0];
u[1] = s[1] | t[1];
endmodule

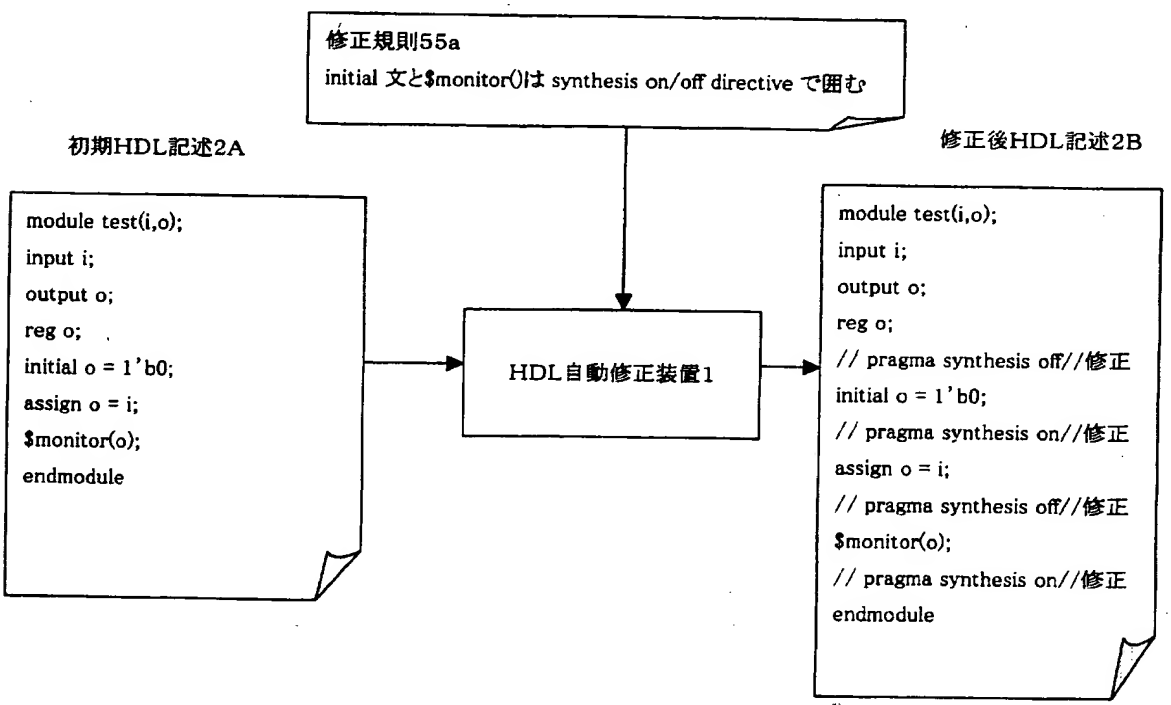
【図７】


修正規則53c
component port: complement to entity;
初期HDL記述2A
library IEEE;
use IEEE.std_logic_1164.all;
entity TEST1 is
port (A, B : in std_logc;
C : out std_logc);
end TEST1;
architecture SAMPLE of TEST1 is
component TEST2
port (S,T : in std_logic;
U : out std_logic);
end component;
begin
T2: TEST2 port map(S=>A,
T=>B,U=>C);
end;
HDL自動修正装置1
修正後HDL記述2B
library IEEE;
use IEEE.std_logic_1164.all;
entity TEST1 is
port (A, B : in std_logc;
C : out std_logc);
end TEST1;
architecture SAMPLE of TEST1 is
component TEST2
port (S,T : in std_logic;
U,V : out std_logic);//修正
end component;
begin
T2: TEST2 port map(S=>A,
T=>B,U=>C,V=>OPEN);//修正
end;
library IEEE;
use IEEE.std_logic_1164.all;
entity TEST2 is
port (S,T : in std_logic;
U,V : out std_logic);
end TEST2;
architecture SAMPLE of TEST2 is
begin
U <= S and T;
V <= S or T;
end;
library IEEE;
use IEEE.std_logic_1164.all;
entity TEST2 is
port (S,T : in std_logic;
U,V : out std_logic);
end TEST2;
architecture SAMPLE of TEST2 is
begin
U <= S and T;
V <= S or T;
end;

【図８】


修正規則53d
port connection: name ;
初期HDL記述2A
module test1(a,b);
input a;
output b;
assign b=a;
endmodule
module test2(p,q);
input p;
output q;
test1 test1_ins(p,q);
endmodule
HDL自動修正装置1
修正後HDL記述2B
module test1(a,b);
input a;
output b;
assign b=a;
endmodule
module test2(p,q);
input p;
output q;
test1 test1_ins(.a(p),.b(q));//修正
endmodule

【図９】


修正規則55a
initial 文と$monitor()は synthesis on/off directive で囲む
初期HDL記述2A
module test(i,o);
input i;
output o;
reg o;
initial o = 1'b0;
assign o = i;
$monitor(o);
endmodule
HDL自動修正装置1
修正後HDL記述2B
module test(i,o);
input i;
output o;
reg o;
// pragma synthesis off//修正
initial o = 1'b0;
// pragma synthesis on//修正
assign o = i;
// pragma synthesis off//修正
$monitor(o);
// pragma synthesis on//修正
endmodule

【書類名】　　要約書

【要約】

【課題】重度の意味論的文法違反を自動修正し且つその修正箇所が明確になるようにして、設計者にかかる負担を大幅に軽減するとともに、高品質のＨＤＬ記述が得られるようにする。

【解決手段】ＨＤＬ字句解析手段１１やＨＤＬ構文解析手段１２のほか、代入文の右辺と左辺の変数の型が不一致になっている部分を意味論的文法違反箇所として検出する意味論的文法違反検出手段１５と、代入文の右辺の変数の型を左辺の変数の型に変換する型変換関数を定義する型変換テンプレート４０と、意味論的文法違反箇所とみなされた代入文の右辺に対して前記型変換関数を適用して意味論的文法違反箇所を正しい記述に修正する意味論的文法違反修正手段１６と、修正後のＨＤＬ記述の逆構文解析を行なう逆構文解析手段１９と、修正箇所に修正に関するコメントを付与するコメント付与手段２０とをそなえて構成する。

【選択図】　　図１

出証特２００１－３０７２２５７

出　願　人　履　歴　情　報

識別番号　　〔０００００５２２３〕

１．変更年月日　　１９９６年　３月２６日
［変更理由］　住所変更
住　所　　神奈川県川崎市中原区上小田中４丁目１番１号
氏　名　　富士通株式会社